POCKET PET

81

81 / 81

POCKET PET NOTIZIE

Redazione:
Harden S.p.A.
Via Pirelli 11
Milano

POCKET PET
anno 1 - numero 2-3
numero unico in attesa
di autorizzazione

Direttore responsabile
Gloriano Rossi

Sommario

Con una lacuna incolmabile	1
Assembler per tutti	2- 9
BASIC & DOS	9
Caccia al ragno	10-13
Tridimensionale	14
Editor MAX	15-20
Alta risoluzione	21-22
Per chi ha l'8000	22
POKE note	23
DEF FNx(X)	24-25
Francobollo	26
Flussi relative	27-32
La Barca Laboratorio	33-34
Pagine>dizero	35-37
Cross Reference	38-42
MODEM 8010	43-45
Unstring CC$ delimited by C$	46-49
Come trasformare, in casa, il PET in 4032	50-52
DOS 1.0 in DOS 2.0	53
Notizie	54
PET show	56-58
In libreria Harden Commodore	59-61
Sparacaratteri	61
ST DS DS$	62-64

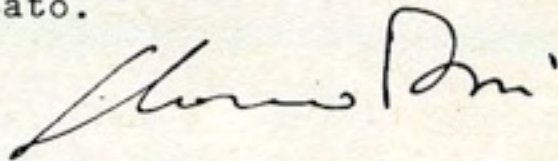
Questo numero di
POCKET PET
e' stato impaginato e
composto in redazione
con l'ausilio di
PET-CBM 3032
PET-CBM 8032
PET-CBM 3022
PET-CBM 8024
PET-CBM 8027
e con :
Wordpro 3.2
Wordcarft 80
VisiCalc e OZZ

Hanno collaborato a questo numero

Alessandro de Simone
Bruno Brazzoduro
Gloriano Rossi
Luciano Odoardi
Massimo Rossi
Paolo Zangrandi
Riccardo Saetti
Roberto Odoardi
Roberto Sozzani

EDITORIALE

CON UNA LACUNA INCOLMABILE

Questo numero di POCKET PET esce come edizione doppia, con valore dei contenuti quadruplo, ma, purtroppo con una lacuna incolmabile.
Questa lacuna e' dovuta alla recente scomparsa del presidente della Harden spa, il geometra Luigi Bonezzi.
E' proprio lui che ha saputo riconoscere, prima di altri, la validita' del personal computer, in particolare del PET.
Sicuro di aver ben riposto la sua fiducia in quel prodotto, di cui tutti Voi siete felici possessori, ha iniziato anni fa la commercializzazione dei prodotti della Commodore.
Luigi Bonezzi era senza dubbio un uomo fuori dal comune, un uomo pieno di umanita', di vitalita' ed intelligenza.
E' proprio con questo senso di umanita' e con quel senso di amicizia che sapeva infondere, che trasformava ogni suo collaboratore in una delle colonne della Harden.
Proprio per questo suo saper sceglere, saper dare ed infondere fiducia, saper dirigere in una atmosfera, si puo' ben dire, famigliare una azienda della portata della Harden, ha fatto si' che questa non perdesse, con la sua mancanza, quella grinta, quella potenza, quella vitalita', quella validita' che egli ha saputo e voluto costruire.o
E' proprio in questa luce, che tutti i dipendenti e collaboratori della Harden, sia pure con una spina nel cuore, proseguono con maggior forza allo sviluppo ed al raggiungimento di quelle mete che lui aveva giustamente impostato.

di Alessandro de Simone.

SSEMBLER PER TUTTI

Prima di continuare il discorso sul set di istruzioni del 6502, vediamo di chiarire alcuni concetti gia' accennati sul numero scorso.

1-Il calcolatore "ragiona" solo in binario puro, tratta cioe' solo gruppi di otto stati di tensione elettrica alla volta alta o bassa, detti BIT.

2-Per semplicita' (vedi fig.2 n.1) il dato formato da otto BIT (detto parola o Byte) viene "spezzato" in due da quattro Bit (detti ciascuno Nibble) e "tradotto" in esadecimale al solo scopo di rendere semplice la vita al programmatore.

3-Per semplificare ancora di piu' la stesura di un programma in LM si ricorre spesso ad un linguaggio detto "Assembler" che, utilizzando gruppi di lettere derivate dalle iniziali delle parole inglesi che indicano la funzione della istruzione, ed incolonnati uno dopo l'altro, consente una relativa facilita' nell'individuare gli eventuali errori o nell'apportare modifiche (fig. 3 n.1).

Esempi:

```
LM        A9 00 8D 00 80 60

Ass.      @$8000=VIDEO : la locazione 8000 (esa) e' definita come
                         "video"
             LDA# #0   : carica l'accumulatore con il dato che segue
                         immediatamente (cioe' zero)
             STA;VIDEO : trasferisci il valore dell'accumulatore nella
                         locazione definita all'inizio come VIDEO,
                         cioe' 8000
             RTS       : return
```

Come si puo' notare nel linguaggio Assembler non e' necesssario ricordare a memoria tutte le istruzioni del 6502 sotto forma di coppie di valori esa ne' tantomeno indicare volta per volta certe locazioni di memoria con l'indirizzo in esa.
D'ora in poi scriveremo i programmi in LM da inserire nel PET tramite il monitor (TIM) con a fianco una "traduzione" in Assembler. Dobbiamo ricordare infatti che ancor piu' che nel caso del BASIC non esiste un Assembler universale, ma vi sono in commercio diversi tipi di Assembler che differiscono l'uno dall'altro in alcuni particolari.

Per chi lo desiderasse e' disponibile su nastro-cassetta, un programma che consente di compilare (tradurre da linguaggio mnemonico in linguaggio macchina) in Assembler. corredato da ampie istruzioni, che potra' girare su tutti i tipi di PET (2000-3000-4000 e 8000) anche vecchie ROM sia pure da 8K e visualizza il programma assemblato su video e su stampante. Esso potra' essere richiesto direttamente all'autore di questa serie di articoli inviando la modesta somma di lire 25000 comprensive di spese di spedizione, di cassetta e documentazione varia.
Il programma suddetto non e' indispensabile, ne' lo sara' per le puntate future, al fine di seguire i programmi che verranno descritti sul POCKET PET; potra' essere utilissimo quale compendio per i lettori che vorranno stendere programmi per conto proprio o per scrivere quei programmi pubblicati da altre riviste che non riportano anche il cosi' detto "codice oggetto", cioe' il programma scritto direttamente in LM.

Calcolo
esadecimale, decimale, binario

Continuiamo ora il discorso sull'LM trattando la corrispondenza esistente tra numerazione esa e decimale. Per esempio per noi il numero 183 significa un numero di oggetti pari alla somma di un centinaio, otto decine e tre unita'. Volendo esprimere questo numero come un insieme di potenze di dieci, noi scriviamo :
183 = 1 x 10↑2 + 8 x 10↑1 + 3 x 10↑0
Si ricorda che qualsiasi numero, tranne 0, elavato alla potenza nulla fornisce come risultato il numero 1.
Inoltre si ricorda che 10↑n e' uguale a 10 x 10 x .....10 n volte. Per esempio 10↑4 = 10 x 10 x 10 x 10 = 10000.
Qualsiasi numero pertanto viene da noi rappresentato come la somma di potenze di dieci, decrescenti (2,1,0 nel nostro caso perche' 183 e' composto di tre cifre) ciascuna moltiplicata per un fattore che e' una delle cifre del numero considerato.

Altri esempi:

Potenza di 10	3	2	1	0	-1	-2	Significato
Valore	8	6	4	8			8x10↑3+6x10↑2+4x10↑1+8x10↑0
			4	2			4x10↑1+2x10↑0
			1	8	,1		1x10↑1+8x10↑0+1x10↑-1
			1	0	,0	2	1x10↑1+0x10↑0+0x10↑-1+2x10↑-2

Tale sistema di numerazione e' stato adottato dall'uomo perche' probabilmente quando scopri' i numeri si servi' del metodo piu' semplice di cui potesse disporre: le dita delle mani.
Un calcolatore, e quindi anche il PET, ha a disposizione solamente uno stato alto o basso a seconda se in un particolare punto del circuito elettrico vi e' tensione o meno. Indicando lo stato alto di tensione con 1 e quello basso con 0, il computer dispone di un sistema che ha appena due simboli e prende appunto il nome di 'SISTEMA BINARIO'.
Una cifra, in tale sistema, prende il nome di bit, ed una quantita' qualunque deve venire espressa come potenza di due.
Concettualmente i due sistemi di numerazione, dec e bin, sono identici, solo che quello dec ha a disposizione 10 simboli, quello binario solamente due, e percio' per indicare una stessa quantita' saremo costretti ad usare piu' simboli, ricorrendo al bin, di quanti ne occarrono usando il dec.

Esempi:

Potenza di 2	3	2	1	0	Significato
Valore		1	1	0	1x2↑2+1x2↑1+0x2↑0 = 5 (dec)
			1	1	1x2↑1+1x2↑0 = 3 (dec)
	1	1	1	1	1x2↑3+1x2↑2+1x2↑0 = 15 (dec)
				0	0x2↑0 = 0

Come abbiamo visto per rappresentare la quantita' quindici sono sufficienti due simboli (1 e 5) nel sistema decimale e ben quattro (1 1 1 1) in quello binario.
Per quantita' non molto piu' grandi (dell'ordine del milione) c'e' bisogno di decine di simboli binari contro i sette del decimale.
Come memorizzarli pertanto in un calcolatore senza spreco di memoria? Il metodo e' relativamente semplice se si ricorre al sistema esa. Dalla figura 1 della scorsa puntata si puo' immaginare facilmente che una certa quantita' sara' rappresentata da un numero di simboli esa inferiore a quello decimale a sua volta nettamente inferiore al binario puro. Poiche' la memoria di un calcolatore e' una sucessione di gruppi di otto bit ciascuno, noi protremo considerare, per semplicita', ciascun byte come se fosse formato da due valori esa benche' nella realta' il valore sia 'scritto' in binario puro (fig. 2 num.prec). Si presenta ora un altro problema nel progetto di un calcolatore, e cioe' come individuare, tra le tante, una certa locazione di memoria. Ad ogni byte si assegna un indirizzo che altro non e' se non un gruppo di sedici bit.

INDIRIZZO H A	INDIRIZZO L A	DATO	TRAD. ESA IND.	TRAD. ESA DATO
0000 0000	0000 0000	0100 0000	0000	40
0000 0000	0000 0001	0100 0010	0001	42
0000 0000	0000 0010	1111 1111	0002	FF
....	....	....	....	..
1010 0010	0000 1100	1100 0110	A20C	C7

In tale modo si possono indirizzare 2↑16=65536 locazioni di memoria. Con un numero inferiore di bit (ad esempio otto) si possono indirizzare solamente 2↑8=256 locazioni di memoria, insufficienti per un versatile uso di un computer.
Da notare che in generale i microprocessori ad otto bit (6502, 8080, Z80 ecc.) trattano come dati gruppi di otto bits ed hanno come indirizzi gruppi di sedici bits.
Per poter gestire una memoria piu' grande, dato che in teoria non c'e' relazione tra numero di bit di un dato e numero di bit di un indirizzo, si potrebbero avere dati di otto bits, ma indirizzi di trentadue e comunque piu' di sedici bits.
Come mai quindi non si scegle questa soluzione per aumentare la capacita' di memoria e invece si ricorre a memorie di massa come i nastri magnetici, i floppy disc, o alle tecniche piu' sofisticate tipo quelle delle memorie virtuali? Semplicemente perche' sarebbe in un primo caso piu' complesso, come vedremo in seguito, indirizzare oltre il valore 65536; in un secondo caso risulterebbe necessario cambiare l'architettura stessa della CPU e di conseguenza modificare la struttura dei linguaggi già' diffusi sul mercato internazionale.
Ritornando agli indirizzi ed ai dati, vediamo ora di individuare un indirizzo di cui sappiamo il valore solo in decimale o esadecimale.
Il problema si presenta quando si usano i comandi PEEK e POKE, dato che siamo costretti a fornire gli argomenti dei comandi stessi espressi come valori decimali, mentre spesso li conosciamo come valori esa.

Conversione di un numero da un sistema ad un altro

Vogliamo convertire il numero 11052 dec nel corrispondente esa (che ha sedici simboli).
Si divide il numero in oggetto per il numero di simboli e si considera il resto ed il quoziente intero. (Primo resto = 12; promo quoziente = 690). Il quoziente, se maggiore o uguale a sedici, si divide nuovamente per sedici (secondo resto 2; secondo quoziente 43); poiche' il nuovo quoziente e' ancora maggiore di sedici si ripete il procedimento finche' si ottiene un quoziente minore di sedici. (terzo resto = 11; terzo quoziente = 2).
Il numero esa desiderato e' formato dai tre resti ottenuti e dall'ultimo quoziente in ordine inverso: 2 11 2 12.
Infine, sostituendo tali valori con i simboli corrispondenti in esa (fig.1 n.p.) si ottiene il valore cercato: 2B2C.
Convertiamo ora un numero esa in dec trattando solamente numeri compresi tra 0000 e FFFF: la prima cifra rappresenta il numero moltiplicato per 16]3; la seconda per 16]2 poi 16]1 e quindi 16]0; pertanto volendo convertire 2B2C si scrive:

2x16↑3+Bx16↑2+2x16↑1+Cx16↑0

cambiando i numeri B e C esa in dec otteniamo:

2x16↑3+11x16↑2+2x16↑1+12x16↑0 = 11052

PEEK e POKE

Prima di continuare e' utile saper usare correttamente le istruzioni BASIC PEEK e POKE.
Queste istruzioni consentono di leggere (PEEK) in tutta la memoria e di scrivere (POKE) un qualsiasi numero intero compreso fra 0 e 255 in qualsiasi locazione della RAM. Per esempio se noi battiamo:
PRINT PEEK (4080)
apparira', in decimale, il valore della 4080ma locazione RAM. Viceversa battendo
POKE 4080,151
il valore 151 dec. sara' trascritto nella 4080ma locazione di memoria.
Da notare pero' che, mentre il comando PEEK(X) legge un valore ed in nessun caso lo modifica, l'istruzione POKE X,Y cerca di modificare il valore della locazione X. Possono infatti verificarsi alcuni casi critici:

1-Cerchiamo di scrivere in una locazione ROM del BASIC o del O.S. (Operative System ; sistema operativo del computer); naturalmente il dato che cerchiamo di scrivere non viene scritto in quanto nelle ROM non si puo' scrivere e nessun messaggio di errore appare per informarci dell'impossibilita' di eseguire l'istruzione.
2-Analoga mancanza di messaggio di errore si verifica se l'indirizzo della POKE cade in una zona RAM non esistente nella configurazione, perche' il sistema, per esempio da 8K, non e' appunto espanso al massimo. In un sistema da 8K l'ultima locazione utilizzabile e' la 8191ma.
3-L'indirizzo della POKE rappresenta una locazione RAM utilizzata dall'O.S. e una sua modifica puo' 'distruggere' il sistema; saremo, in questo caso, costretti a spegnere e poi riaccendere il computer, perdendo purtroppo tutto il contenuto delle RAM.
4-Il valore Y di POKE X,Y e' negativo o e' maggiore di 255. Questo e' l'unico caso in cui appare un messaggio di errore. Teniamo presente infine che se Y e' un valore non intero verra' presa in considerazione solamente la parte intera.

Supponendo allora di non cadere in uno dei casi critici, come facciamo a memorizzare un numero maggiore di 255? Il procedimento e' un po' lungo, ma e' l'unico che sia possibile adottare:
Vogliamo memorizzare il numero intero 11052 dec. Abbiamo gia' visto che in esa corrisponde a 2B2C, e poiche' ogni cifra esa rappresenta quattro bits, in totale avremo bisogno di sedici bits. Dato che ogni locazione di memoria contiene otto bits noi potremo 'spezzare' 2B2C in due gruppi: 2B e 2C; il primo, dopo averlo tradotto in dec., lo scriveremo in una locazione, il secondo in quella successiva.

Riassumiamo il procedimento in altre parole:

```
11052 dec. = 2B2C esa
2B     2C
2B esa = 2x16↑1+Bx16↑0 = 2x16+11 = 43 dec.
2C esa = 2x16↑1+Cx16↑0 = 2x16+12 = 44 dec.
```

Utilizziamo, per le verifiche che seguono, le locazioni RAM da 826 dec (033A esa) fino alla 1023ma, perche' tali locazioni sono destinate alla gestione della seconda cassetta e non rischiamo di distruggere il sistema.
Scriviamo:
POKE 826,44: POKE 827,43
Come si puo' notare, del numero 2B2C abbiamo trascritto nella prima locazione (826) il valore 2C (LSB : Least Significant Byte = Byte meno significativo) e nella successiva (827) il 2B (MSB : Most significant Byte = Byte piu' significativo) anziche' al contrario come saremmo stati indotti a fare istintivamente.
Adottiamo questo procedimento perche', come abbiamo visto a pagina 3 della prima puntata, lo segue anche la CPU.
Come faremo, allora, a sapere quale numero e' rappresentato da due locazioni di memoria successive? Naturalmente seguendo il ragionamento inverso:
Traduciamo l'MSB in decimale e lo moltiplichiamo per 256
2B esa = 43 dec; 43 x 256 = 11008
Analogamente ci comportiamo per l'LSB moltiplicandolo pero' per 1 (lasciandolo cioe' invariato):
2C esa = 44 dec
In seguito eseguiamo la somma fra i due:
11008 + 44 = 11052

Altri esempi:

	LSB	MSB	
Loc.RAM	826	827	Significato
	10	15	15x256+10 = 3850
	112	100	100x256+112 = 25712
	1	88	88x256+1 = 22529
	1	1	1x256+1 = 257
	255	0	0x256+255 = 255
	255	255	255x256+255 = 65535 (?)

Come si puo' notare il numero piu' grande che si puo' rappresentare e' 65535 ed il piu' piccolo e' zero, ma tutti positivi. Se pero' rinunciamo ad ottenere valori cosi' grandi possiamo seguire la convenzione secondo cui sono da considerare positivi i valori fino al numero (65535-1)/2 cioe' da zero a 32767, mentre da 32768 a 65535 inclusi sono negativi e valgono esattamente il valore considerato meno 32767:

Esempi:

```
52768  significhera'  -(52768-32767).= -20001
65535  significhera'  -(65535-32767) = -32768
32768  significhera'  -(32768-32767) = -    1
```

In questo modo possiamo rappresentare tutti i numeri interi negativi e positivi compresi fra -32768 e +32767. Ecco spiegato, dunque, perche' certi computers non molto sofisticati (ed anche il PET quando usiamo variabili intere tipo A%) trattano solamente quei numeri interi: con due bytes adiacenti non e' possibile superare l'intervallo suddetto.

Somma fra numeri esa

Siamo ora in grado di 'scrivere' un numero intero compreso tra -32768 e +32767. Vediamo ora come eseguire la somma fra due numeri dec e poi esa.
Facendo la somma dell'esempio qui sotto riportato, notiamo che quando la somma di due cifre corrispondenti supera il numero nove, dobbiamo considerare il riporto nella successiva colonna (somma tra 8 e 5; tra 6 e 4).

Decimale	Esadecimale
1 1 Riporto	1 Carry
16280+	1A03+
4250=	9B77=
20530	B57A

Analogamente avviene per due numeri esa, solo che consideriamo il riporto quando la somma supera il numero F (15); (es.: somma fra A e B della figura sopra riportata).
Prima di continuare ricordiamo che in inglese riporto si traduce con il termine CARRY, ed in seguito lo chiameremo sempre in questa maniera.
Tra le istruzioni in LM del 6502 ve ne sono ben otto di somma tra due bytes e ne esaminiamo ora una:

Codice esa 69: istruzione a due bytes; codice mnemonico ADC# derivato da ADd accumulator immediate with Carry.
Esempio:

```
LM : 69 72

Ass :    ADC# #$72
```

Quando il uP incontra questa istruzione, esegue la somma tra il valore che si trova in quel momento nell'accumulatore, il numero esa 72 e l'eventuale Carry.

Vediamo ora alcuni esempi:

```
   LM               ASSEMBLER

                    @826 = START
033A 18             CLC
033B A9 02          LDA# #2
033D 69 07          ADC# #7
033E 8D 45 03       STA ; QUI
0341 60             RTS
.... EA             NOP
.... EA             NOP
     EA             NOP
     EA                  = QUI
     EA             NOP
     EA             NOP
.... ..             ...
```

REMarks

033A 18 CLear Carry (CLC)
Istruzione ad un solo byte, implicita, cioe' non ha bisogno di indirizzi o di dati per la sua interpretazione. Questa istruzione cancella il Carry eventualmente presente nel registro della CPU. Ritorneremo in seguito su questa istruzione.

033B A9 02   LDA# #2   vedi num. prec.

0341 60   RTS Return subrutine; istruzione implicita anche questa: non c'e' bisogno di altre informazioni per specificare il compito da svolgere.

0342 EA ... EA NOP No OPeration. Trascriviamo questo gruppo di EA perche' in seguito sia piu' semplice rintracciare sul video il risultato della somma. Per la cronaca EA e' una istruzione implicita che significa : 'non eseguire alcuna operazione'; al contrario di quanto possa sembrare e' una istruzione molto utile, come vedremo in seguito.

Per fare girare il programma battiamo di seguito:
POKE 828,2: POKE 830,7: SYS (826) e poi chiamiamo il monitor (TIM) mediante SYS(4) (nuove ROM) o SYS(1039) (vecchie ROM, dopo aver caricato il TIM). Apparira':

```
     PC   IRQ  SR AC XR YR SP
.;  0005 E455 30 00 5E 04 F8
.
```

battiamo: M 033A 034A   Apparira':

```
.M 033A 034A
.;  033A 18 A9 02 69 07 8D 45 03
.;  0342 60 EA EA 09 EA EA EA ..
.X
READY
```

Ora sia modificando il monitor sia ricorrendo a POKE 828,nn e POKE 830,nn , modifichiamo il valore di 033C e di 033E e ripetendo le operazioni precedenti, in 0345 comparira' sempre il risultato della somma, a meno che la somma dei due numeri caricati non superi il valore FF; infatti consideriamo le seguenti somme esa:

```
          0C + 04 = 10
          40 + 0A = 4A
          0B + 03 = 0E
invece :  FC + 14 = 10
          AB + 63 = 0E
```

Infatti:

```
      0C = 12 dec

      04 = 04 dec

04 + 12 = 16 dec = 10 esa
```

Come si puo' notare allo stesso risultato si puo' arrivare anche sommando FC e 14. Come facciamo ora a sapere se il valore contenuto in 0345 e' il risultato di una somma inferiore o superiore ad FF?

A tale scopo consideriamo il registro di stato (ST) della CPU 6502: esso e' un particolare registro di sedici bits interno alla CPU stessa che memorizza diverse informazioni tra le quali il Carry: Il bit dell'ST corrispondente al Carry viene automaticamente posto (settato) ad 1 se, dopo l'esecuzione di alcune istruzioni, tra le quali la somma, si supera il valore FF. Da quel momento esso non viene piu' cancellato (cioe' resettato o portato a 0) anche se in seguito si esegue una somma che non ha riporto!
Ecco perche' la prima istruzione di una procedura di somma deve SEMPRE essere CLC che viene cosi' interpretata: resetta il Carry che eventualmente e' stato posto ad 1 da una operazione precedente.
Miglioriamo ora il programma in LM visto e, per fare questo, esaminiamo un'altra istruzione di BRANCH (salto condizionato):
BCS : Branch on C Set to 1 = Salta se il Carry e' uguale ad 1.
Codice operativo : B0 xx; dove xx rappresenta l'entita' del salto, vedi pag.5 numero scorso (REMarks 034C).

```
               @826 = START
033A 18        CLC
     A9 00     LDA# #0
     8D 52 03  STA; RIPORTO
     A9 02     LDA# #2
     69 07     ADC# #7
     8D 45 03  STA; BYTE SOMMA
     B0 01     BCS + MODIFICA
     60        RTS
     A9 01     = MODIFICA
               LDA# #1
     8D 52 03  STA; RIPORTO
     60        RTS
     EA        NOP
.... ..        ...
     EA        = RIPORTO
     EA        = BYTE SOMMA
     EA        NOP
```

REMarks

A9 00
8D 52 03 Queste due istruzioni scrivono 00 nella locazione 0352 che servira' a visualizzare l'eventuale riporto.

B0 01 Se la somma appena eseguita supera il valore FF il bit di Carry viene posto a 1 e pertanto il salto di un byte e' eseguito e l'elaborazione continua da 034A.

60 Se la somma eseguita non ha riporto il Carry rimane come prima, e cioe' a 0 (cfr. ist. 033A 18); si ritorna al BASIC.

A9 01
8D 52 03 Nella locazione 0352 indicata in Assembler con RIPORTO viene trascritto il valore 01.

Facciamo ora girare il programma nel solito modo (SYS 826) ricordando che i nuovi indirizzi delle POKEs sono 833 e 835. Se la somma eseguita avra' riporto o meno lo noteremo dalla presenza, o meno, di 01 in 0352.

Dato che ci siamo commentiamo alcune righe del programma scritto in Assembler.

@826 = START Comunica al programma ASSEMBLER, definendola con il nome 'START', la locazione dalla quale il programma LM deve essere allocato. Il programma (ASSEMBLER) automaticamente. provvedera' ad incrementare gli indirizzi per le successive istruzioni. Cio' significa che se per esempio vogliamo collocarlo nella memoria a partire dalla locazione 0355 invece che dalla 033A sara' sufficente modificare solamente @$0355=START; a modificare tutti gli indirizzi provvedera' automaticamente l'ASSEMBLER.

STA; RIPORTO. Memorizza il valore attuale (corrente) dell'accumulatore nella locazione indicata in seguito con 'riporto'.

STA; BYTE SOMMA. Trasferisce il valore corrente dell'accumulatore nel byte definito in questo modo.

BCS + MODIFICA. Se il Carry e' uguale a 1 il programma continua dalla locazione indicata come 'MODIFICA'. Attenzione: l'ASSEMBLER elabora automaticamente l'entita' del salto evitando noiosi calcoli!

I vari nomi di fantasia che facilitano nettamente la stesura di un programma sono chiamati LABELS (etichette).
Come si puo' notare il programma scritto in Assembler e' di gran lunga piu' semplice da interpretare dell'LM.
Eseguiamo ora una somma di due numeri, ciascuno dei quali occupa due bytes.

11290+
14891=

26181

In pratica dobbiamo dapprima eseguire la somma tra gli ultimi due bytes (EA + 21), trascrivere il risultato da qualche parte e tener presente l'eventuale Carry (nel caso specifico esiste). In seguito sommiamo i primi due bytes fra loro e l'eventuale Carry (2C + 32 + 1).
Il programma puo' essere il seguente:

```
CLC
LDA# #$EA
ADC# #$21
STA;LSB
LDA# #$2C
ADC# #$32
STA;MSB
RTS
=LSB
=MSB
```

Prima di eseguire la seconda somma (LDA# #$2C - ADC# #$32) NON bisogna inserire CLC perche' e' necessario considerare il Carry eventualmente presente in seguito alla prima somma.
Il lettore, per esercizio, puo' modificare il programma inserendo una 'spia' per l'eventuale secondo riporto, come abbiamo fatto nel programmino precedente.
Prima di terminare parliamo di un'altra istruzione implicita: SED (SEt Decimal mode), codice operativo F8.
Dall'istruzione successiva a SED le somme vengono eseguite in decimale anziche' in esa! F8 e' di notevole comodita' in molti casi: provate ad inserirla nei programmi precedenti prima di CLC o subito dopo. Attenzione pero' che se il modo di operare e' decimale e chiediamo di sommare valori esa si verificano degli errori.
Per ripristinare il modo esa e' necessaria l'istruzione implicita CLD (CLear Decimal mode) codice operativo D8; senza questa istruzione infatti la CPU continuera' sempre a ragionare in decimale. Consiglio a chi desidera sperimentare l'istruzione SED di inserire D8 prima di ogni RTS per evitare malfunzionamenti del BASIC.

Per concludere, a partire dalla locazione 033A, inserite il seguente programma e battete SYS(826) dopo aver cancellato lo schermo.

```
.: 033A A2 00 BD 48 03 9D 00 80
.: 0342 E8 E0 18 D0 F5 60 2A 2A
.: 034A 2A 2A 2A 2A 2A 2A 2A 2A
.: 0352 10 0F 03 0B 05 14 2A 2A
.: 035A 10 05 14 2A 2A 2A 2A 2A
.: 0362 2A 2A 2A 2A 2A .. .. ..
```

Dato che contiene istruzioni che non abbiamo ancora incontrato lo commenteremo nella prossima puntata e non vi diciamo a cosa serve per incuriosirvi. Il lettore tenga comunque presente che 033B contiene l'inizio; 0343 contiene la fine; da 0348 a 036... si puo' modificare il contenuto senza alcun pericolo... che cosa aspettate a decifrare quest'ultima frase?.

Il recapito dell'autore di questa serie di articoli e' il seguente:

Alessandro de Simone
casella postale 74
20035 Lissone (Milano)

--*H*-----*H*-----*H*--

BASIC & DOS

Con questo breve programma si puo' riconoscere la versione BASIC e a quale DOS appartengono i nostri floppy disk.
Opportunamente scelte, le istruzioni di riconoscimento (righe 230 fino 290), possono essere adottate in qualsiasi altro programma gestionale e non.
Una piccola utility, dunque, per rendere i vostri programmi universali.

```
100 REM*********************************
110 REM* RICONOSCIMENTO AUTOMATICO     *
120 REM* VERSIONE PET-CBM E FLOPPY     *
130 REM* ---------------------------   *
140 REM* GLORIANO ROSSI - USER GROUP*
150 REM*********************************
160 B$(0)="SERIE 2001 BASIC 1.0"
170 B$(1)="SERIE 3000 BASIC 2.0"
180 B$(2)="SERIE 4000 BASIC 4.0"
190 B$(3)="SERIE 8000 BASIC 4.0"
200 D$(1)="FLOPPY DISK 2040 (DOS 1.0)"
210 D$(2)="FLOPPY DISK 4040 (DOS 2.0)"
220 D$(3)="FLOPPY DISK 8050 (DOS 2.5)"
230 A=PEEK(57345):TP=0:IFATHENTP=1:IFAAND1THENTP=3:IFAAND4THENTP=2
240 OPEN15,8,15
250 PRINT#15,"M-R"CHR$(255)CHR$(255)
270 GET#15,A$
280 CLOSE15
290 A=ASC(A$):TD=1:IFAAND16THENTD=3:IFAAND1THENTD=2
300 REM************ RISULTATI
310 PRINTB$(TP):PRINTD$(TD)
```

Caccia al ragno

Questo programma,come dice il titolo,vi permettera' di cimentarvi in una appassionata quanto pericolosa caccia al ragno.

di
Roberto Sozzani

Prima di esaminare per somme linee il programma, vediamo un po' le regole del gioco: sulla sinistra dello schermo apparira' la canna del vostro fucile, ad una altezza variabile scelta casualmente.
Dall'alto scendera' un ragno, calandosi con la sua ragnatela.
Posizione sullo schermo e velocita' di discesa sono anch'esse scelte casualmente.
Per sparare bisogna premere il tasto "space", tenendo presente che il ragno si arresta al momento dello sparo.
Se si colpisce il ragno, tutto bene; altrimenti cominciano i guai!
Infatti questi sono ragni di una razza particolare, molto intelligente. Se non li colpite, questi furbacchioni vanno a costruire un muro, piazzando un mattone alla volta, proprio davanti al vostro fucile.
Piu' il muro diventa alto, piu' difficile diventera' per voi poter sparare.
Potra' capitare, infatti, che il fucile venga posizionato ad una altezza inferiore a quella del muro.
In questo caso il ragno potra' scendere indisturbato, e sistemare un altro mattone sul muro, aumentando ulteriormente il vostro handicap.
Anche per voi pero' c'e' una possibilita': i vostri proiettili, di una lega speciale, sono in grado di sgretolare un mattone in due colpi. Il primo colpo comprime il mattone, mentre il secondo lo attraversa come se niente fosse. Se vi trovate di fronte il muro, quindi, sparate lo stesso! Se in futuro vi troverete nella stessa posizione, potrete sparare al ragno forando il muro. Ma attenzione: il prossimo ragno che non colpirete, andra' a tappare il buco, invece di piazzare il mattone sulla sommita' del muro.
I ragni in totale sono 20, ed a turno scenderanno per costruire il muro. Se li uccidete tutti, avete vinto; se invece il muro raggiunge una certa altezza, tale da non permettervi piu' di sparare, allora i ragni prendono il sopravvento e voi avete perso. In alto a destra apparira' il numero dei ragni uccisi, a sinistra il numero dei colpi sparati. Se volete interrompere il gioco, ricordatevi di premere il tasto "Q" e non "RUN STOP" (poi vi spieghero' il perche').
A questo punto non vi resta che copiare il programma e poi...comincia la strage !!

REMarks

RIGHE 10, 20, 930, 940, 950, 960, 970: breve routine in linguaggio macchina che crea una cornice sullo schermo.
Chi disponesse di PET con vecchie ROM, dovra' apportare le seguenti modifiche:

I) Cancellare le righe sopraindicate;
II) inserire le seguenti righe:

```
42 FORI=32768TO32807:POKEI,102:NEXT
44 FORI=32808TO33728STEP40:POKEI,102:POKEI+39,102:NEXT
46 FORI=33728TO33768:POKEI,102:NEXT:RETURN
```

III) Modificare:
Riga 40: sostituire SYS 826 con 'GOSUB 42: GOTO 50'
Riga 260: sostituire SYS 826 con 'GOSUB 42'

RIGHE 40/160: presentano il programma. Il ragno si muove sullo schermo e fa dispetti. Prosegue fino a quando non si preme un tasto a caso.
RIGA 170: regola la velocita' di caduta (V viene determinata alla riga 340).
RIGA 180: regola le pause.
RIGHE 190/220: Regolano il suono; la riga 220 azzera i valori della user port, altrimenti i comandi si SAVE e LOAD restano disabilitati. E' importante uscire dal programma premendo il tasto "Q" e non RUN STOP. Se si preme "Q" infatti si passa automaticamente dalla riga 220 (vedi riga 390).
RIGHE 230/250 - 690/820: istruzioni.
RIGA 270: annulla tutti i tasti premuti dopo il primo sparo.
RIGA 280: azzeramento delle variabili e stampa del numero dei ragni uccisi.
RIGA 290: stampa colpi sparati.
RIGHE 300/320: costruzione canna del fucile, calcolo riga dello schermo e sua stampa.
RIGHE 350/360: calcolo colonna di discesa del ragno.
RIGHE 370/400: discesa del ragno.
RIGHE 420/440: sparo.
RIGHE 450/510: analizza, tramite il comando PEEK, cosa incontra il proiettile nella sua traiettoria.
RIGA 520: se il ragno arriva al pavimento, rimanda alla subroutine di riga 900.
RIGA 530: se il muro ha raggiunto una certa altezza, il gioco finisce e tu hai perso.
RIGA 540: cancella la parte superiore della ragnatela. La variabile TT = SIN (5) serve solo a creare un certo rallentamento nell'esecuzione del ciclo di FOR...NEXT.
RIGHE 550/560: cancella la canna del fucile e si passa ad un nuovo ragno.
RIGHE 570/680: fine del gioco e riassunto dei dati.
RIGHE 830/840: taglio della ragnatela.
RIGA 860: cancella la parte inferiore della ragnatela.
RIGHE 880/890: caduta del ragno dopo il taglio della ragnatela.
RIGHE 900/920: costruzione del muro.

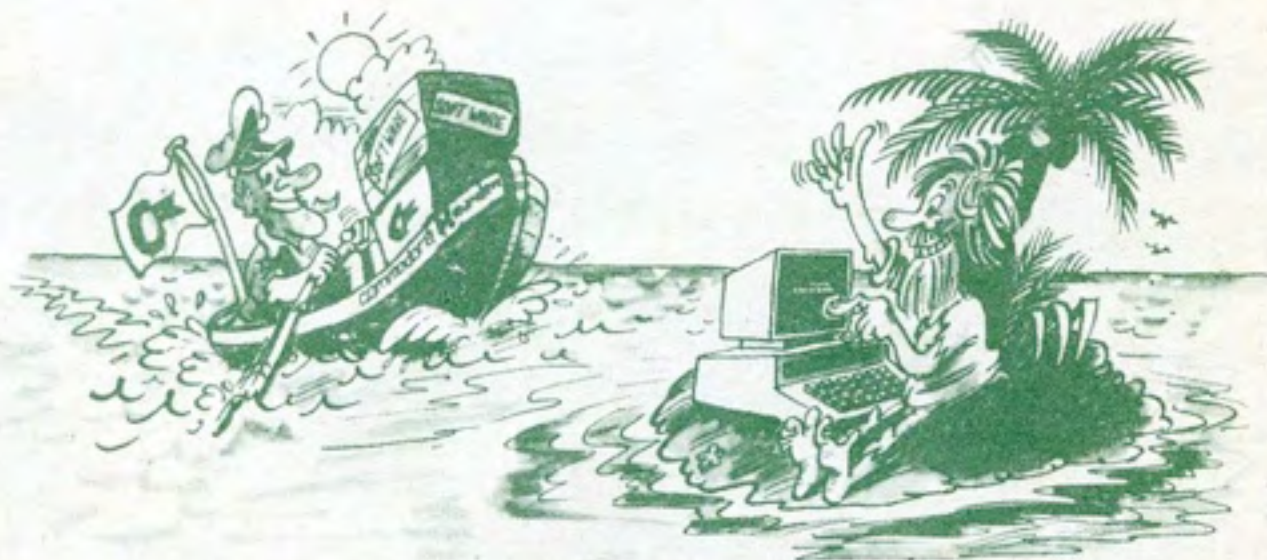
Caccia al Ragno nero

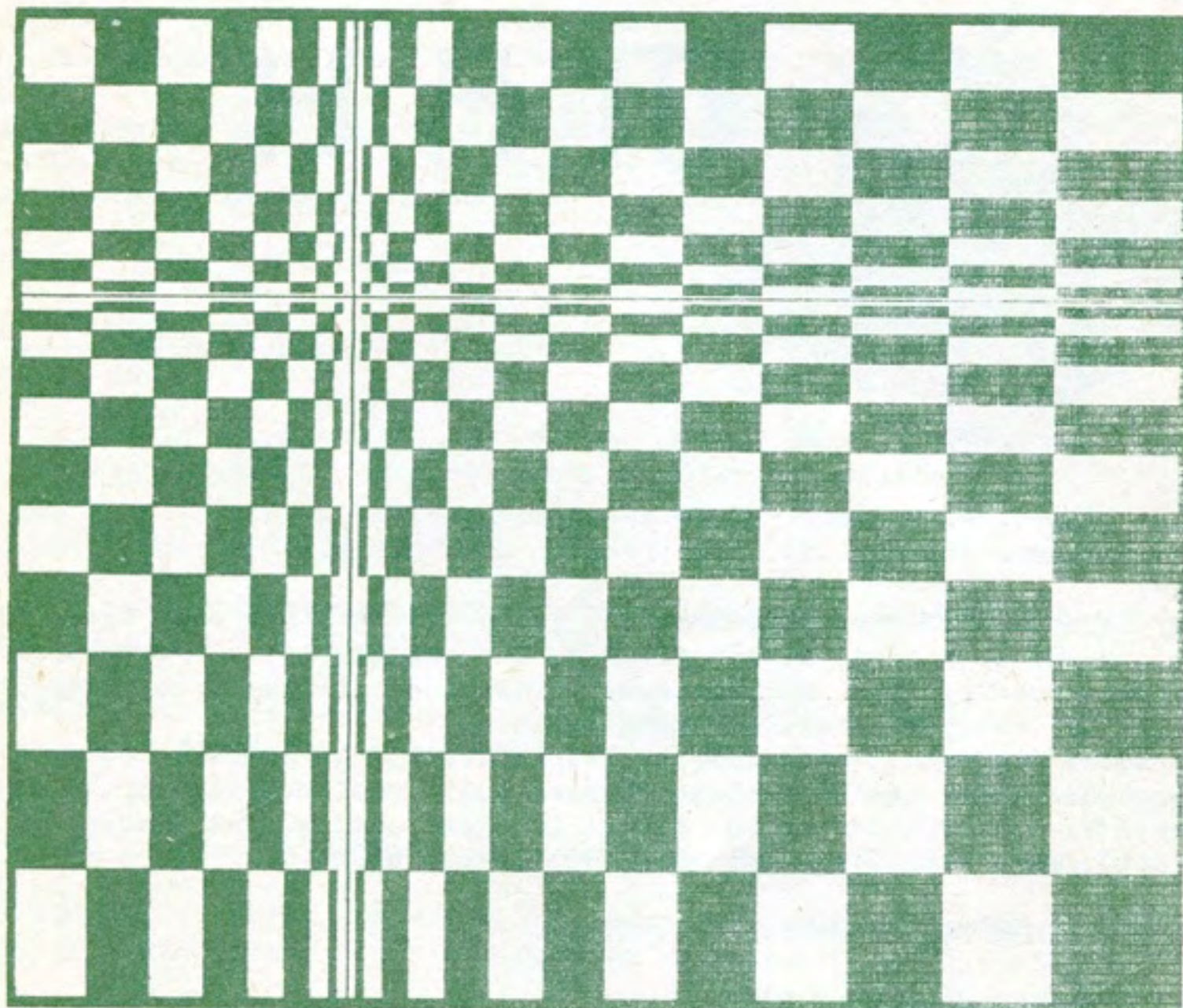
```
1 REM ***************************
2 REM *** (C) ROBERTO SOZZANI ***
3 REM ***     POCKET GROUP    ***
4 REM ***     MILANO 10/2/81  ***
5 REM ***************************
6 REM ***  CACCIA  AL         ***
7 REM ***     RAGNO  NERO     ***
8 REM ***************************
9 REM
```

```
          ***      ***
        *    *    *    *
               ***
      **** ***** ****
    *      *****      *
      ***** *** *****
    *     *    *     *
            *    *
          *        *
```

```
10 READ A,B
20 FORI=ATOB:READX:POKEI,X:NEXTI
30 PRINT"▯"
40 P=59467:O=59466:N=59464:C=51:TR=20:SYS826
50 POKE216,12:POKE198,10:SYS57949:PRINT"CACCIA AL RAGNO (2) !"
60 PRINTTAB(12)"██- POCKET GROUP -"
70 PRINTTAB(13)"█BY EX I2JQ (π)"
80 PRINTTAB(6)"████PREMI UN TASTO PER COMINCIARE"
90 GOSUB180
100 X=32822:POKEX,93:S=40:L=93:M=32:V=200
110 FORK=1TO10:GETA$:IFA$<>""THEN230
120 X=X+S:POKEX,42:GOSUB200:GOSUB170:POKEX,L:NEXT
130 IFL=32THENGOSUB180:GOTO100
140 POKEX,42:GOSUB170:IFPEEK(X+40)=32THENM=9
150 GOSUB170:POKEX+40,M:GOSUB190:GOSUB170:POKEX,32
160 S=-40:L=32:GOTO110
170 FORJ=1TOV:NEXT:RETURN
180 FORJ=1TO1000:NEXT:RETURN
190 POKEP,16:POKEO,85:FORI=250TO1STEP-40:POKEN,I:NEXT:GOTO220
200 POKEP,16:POKEO,C:POKEN,255:FORI=1TO10:NEXT:GOTO220
210 POKEP,16:POKEO,15:FORI=5TO255STEP5:POKEN,I:NEXT
220 POKEP,0:POKEO,0:POKEN,0:RETURN
230 PRINT"▯███████████              ▚VUOI LE ISTRUZIONI ?"
240 GETA$:IFA$=""THEN240
250 IFA$="S"THENGOSUB690
260 PRINT"▯":SYS826
270 FORG=1TO20:GETA$:NEXT
280 SP=0:RA=0:RR=0:C=51:WW=0:POKE216,0:POKE198,35:SYS57949:PRINT"▚"SC
290 PRINT"▚"TAB(3)"▚"T
300 F$="▔▔▔▔█████▔▔▔▔":RI=INT(20*RND(1)+1):IFRI<8THENRI=8
310 F0$="    █████    "
320 POKE216,RI:POKE198,1:SYS57949:PRINTF$
330 GOSUB180
340 V=(4*RND(1)+.5)*30
350 CO=INT(22*RND(1)+1)
360 X1=32818+CO:POKEX1,93
370 FORW=1TO22
380 X1=X1+40:POKEX1,42:IFSP=0THENGETA$:IFA$=" "THENSP=1:GOTO420
390 IFA$="Q"THENGOSUB220:GOTO570
400 GOSUB200:GOSUB170:POKEX1,93:NEXT:POKEX1,42
410 GOTO520
420 T=T+1:X=32811+40*(RI-1):FORQ=1TO36:POKEX+Q,64:POKEX+Q,32
430 IFPEEK(X+Q+1)<>32THEN450
440 NEXTQ
450 IFPEEK(X+Q+1)=102ANDQ<5THENPOKEX+Q+1,118:GOSUB210:POKEX1,93:NEXTW
460 IFPEEK(X+Q+1)=118THENNEXTQ
470 IFPEEK(X+Q+1)=102THENC=51:GOSUB210:IFWW=0THENPOKEX1,93:NEXTW
480 IFRR=1THENGOSUB850:GOTO530
490 IFPEEK(X+Q+1)=42THENC=15:TR=TR-1:GOSUB200:SC=SC+1:WW=1:NEXT
500 IFTR=0THEN630
510 IFPEEK(X+Q+1)=93THENGOSUB190:RR=1:X2=X1:WW=1:GOSUB830:NEXTQ
520 IFPEEK(X1+40)=102THENM=X1:W=22:C=15:GOSUB200:POKEX1-40,75:V=10:GOSUB900
530 IFMM=<33097ANDMM>0THEN570
540 FORI=WTO1STEP-1:X1=X1-40:POKEX1,32:TT=SIN(5):NEXT
550 POKE216,RI:POKE198,1:SYS57949:PRINTF0$
560 GOTO270
```

```
570 PRINT"■■■■■■■I RAGNI HANNO AVUTO IL SOPRAVVENTO !!"
580 IFA$<>"Q"THENPRINT"■NON PUOI PIU' SPARARE!!"
590 PRINT"■HAI UCCISO";SC;"RAGNI";:IFSC=1THENPRINT"■O";
600 PRINT" IN";T;"TIRI";:IFT=1THENPRINT"■O■"
610 PRINT:PRINT"■MA";TR;"RAGNI SONO ANCORA VIVI !!"
620 PRINTTAB(10)"■■■■■■■VUOI RITENTARE ?":GOTO660
630 PRINT"■■■■■■■COMPLIMENTI ! HAI UCCISO TUTTI I RAGNI"
640 PRINTTAB(13)"■IN";T;"TIRI"
650 PRINTTAB(8)"■■■■■■■VUOI GIOCARE ANCORA ?"
660 GETA$:IFA$=""THEN660
670 IFA$<>"N"THENT=0:SC=0:TR=20:MM=0:GOTO260
680 END :REM FINE GIOCO "

690 PRINT"■■LA TUA CASA E' INFESTATA DA RAGNI DI UNA"
700 PRINT"■RAZZA PARTICOLARE MOLTO INTELLIGENTE."
710 PRINT"■AL MOMENTO CE NE SONO 20 ED A TURNO"
720 PRINT"SCENDERANNO DAL SOFFITTO; TU PUOI SPARA-RE PREMENDO ■SPACE■."
730 PRINT"■OGNI VOLTA CHE RAGGIUNGONO IL PAVIMENTO AGGIUNGONO UN MATTONE AD";
740 PRINT" UN MURO CHE     COSTRUISCONO PER RIPARARSI DAI TUOI PRO-IETTILI."
750 PRINT"■SE PERO' COLPISCI DUE VOLTE LO STESSO   MATTONE, LO DISINTEGRI."
760 PRINT"■PER AIUTARTI A TENERE IL CONTO, IN ALTO A DESTRA APPARIRA' IL";
770 PRINT" NUMERO DEI RAGNI  UCCISI, A SINISTRA QUELLO DEI COLPI "
780 PRINT"SPARATI ."
790 PRINT"■PER FINIRE PREMI ■'Q'■."
800 PRINT"■      ■PREMI UN TASTO PER INIZIARE"
810 GETA$:IFA$=""THEN810
820 RETURN :REM RITORNO DOPO LE SPIEGAZIONI"

830 POKEX+Q-39,74:IFPEEK(X+Q+41)=93THENPOKEX+Q+41,85:RA=1
840 RETURN
850 IFRA=0THEN870
860 FORCC=32818+CO+40*RITOX1-40STEP40:POKECC,32:NEXT
870 C=15:V=10:IFW=22THENGOSUB200:RETURN
880 POKEX1,32:FORM=X2TO(X2+(21-W)*40)STEP40:POKEM,42:GOSUB170:POKEM,32:NEXT
890 POKEM,42:GOSUB200
900 POKEM,32:FORMM=MTO33698STEP-1:POKEMM,42:GOSUB170:C=51:GOSUB200:POKEMM,32
910 NEXTMM:FORA=1TO20:IFPEEK(MM-2)<>32THENMM=MM-40:NEXTA
920 POKEMM-1,42:GOSUB190:POKEMM-2,102:GOSUB180:POKEMM-1,32:RETURN:REM"

930 DATA 826, 890
940 DATA 173,121,3,133,1,173,122,3,133,2,162,23,160,0,169,102
950 DATA 145,1,200,192,40,208,249,224,23,208,35,160,0,24,165,1
960 DATA 105,40,133,1,165,2,105,0,133,2,169,102,145,1,160,39
970 DATA 145,1,202,208,230,165,1,105,40,133,1,76,70,3,96,0,128
```

--*H*----*H*----*H*----*H*----*H*--

tridimensionale

Luciano ha colpito ancora! Una ne fa, ma cento ne pensa. Con un PET ed una stampante grafica tipo 3022 o 4022, ecco cosa si puo' ottenere:

```
100 REM ********************      TRIDIMENSIONALE      ********************
110 REM *                   *                         *                    *
120 REM *   POCKET GROUP    ***************************   EFFETTO 3022    *
130 REM *                                                                  *
140 REM ************************************* LUCIANO ODOARDI & FIGLIO *
150 REM
160 A$="▪     ▬▬▬   ▬▬  ▬ ▪¦▪ ▬  ▬▬  ▬▬▬   ▬▬▬▬     ▬▬▬▬▬     ▬▬▬▬▬▬"
170 B$="▬▬▬▬▬    ▬▬▬   ▬▬  ▪ | ▪ ▬▬  ▬▬▬   ▬▬▬▬    ▬▬▬▬▬     ▬▬▬▬▬▬      ▪"
180 OPEN6,4,6:PRINT#6,CHR$(9):OPEN1,4:CMD1:A=7:B=1:C=13:E=64
190 C$="▬":D$=" ▪":E$="▪":GOSUB270
200 FORD=BTOA:PRINTA$:NEXT:A=A-B:IFA<BTHENC$="-":GOSUB270:GOTO230
210 FORD=BTOA:PRINTB$:NEXT:A=A-B:IFA<BTHENC$="-":GOSUB270:GOTO230
220 GOTO200
230 A=B
240 FORD=BTOA:PRINTA$:NEXT:A=A+B:IFA>CTHENC$="▬":GOSUB270:GOTO280
250 FORD=BTOA:PRINTB$:NEXT:A=A+B:IFA>CTHENC$="▬":GOSUB270:GOTO280
260 GOTO240
270 PRINTD$;:FORL=BTOE:PRINTC$;:NEXT:PRINTC$;:PRINTE$:RETURN
280 PRINT#6,CHR$(24):CLOSE6:CLOSE1
```

EDITOR

MAX

di Massimo Rossi

I lettori esperti ci scuseranno, se spendiamo qualche parola allo scopo di chiarire, ai meno esperti, il significato e l'utilita' di un programma "EDITOR".
Si dice, infatti EDITOR un programma in grado di gestire un testo letterario o grafico, e cioe' di modificarlo, di correggerlo, di memorizzarlo, sia temporaneamente, sia su supporto (Cassette, Floppy), e quindi di stamparlo, ed, eventualmente, di impaginarlo.
Vi sono molti programmi, decisamente sofisticati, in genere scritti in Linguaggio Macchina, che permettono di fare tutto cio' in grande stile (vedi, ad esempio il Word Processing, di cui e' la descrizione sul numero 0 di Pocket Pet), ma questi programmi hanno due pseudo difetti:

- esigono una memoria di 32 Kbytes

- vengono gestiti da Floppy Disk

Percio' chi non possiede una Unita' Floppy Disk oppure non dispone di una memoria adeguata, ma possiede una Stampante, non puo', neppure volendo, usufruire di questi programmi.
Abbiamo percio' deciso di realizzare un programmino che avesse le caratteristiche fondamentali di un Editor, pur essendo molto breve, ed essendo scritto in Linguaggio Basic.
Ci siamo messi al lavoro, ed abbiamo ottenuto un programma che, nella sua brevita', ha delle sorprendenti caratteristiche.
Esso, infatti, permette di usare lo schermo del PET come la pagina di un quaderno, dove si puo' scrivere, o disegnare, cio' che si desidera.
Se vi sono degli errori, questi possono venire corretti, usando i tasti che comandano il cursore (come se si stesse correggendo un programma). Una volta otenuto il testo e giudicatolo soddisfacente, si puo' memorizzare l'intera pagina, collocandola in un zona di memoria libera del PET, e si puo' creare un numero crescente di pagine in proporzione alla memoria a disposizione dell'apparecchio (il programma stesso esegue questo Test, e percio', non vi permette di sbagliare).

Quando si e'finito di memorizzare, si puo' richiamare la pagina che si desidera, rivederla, e, se si desidera, stamparla.
Se il testo viene ritenuto degno di essere conservato, lo si puo' registrare, pagina per pagina, su Cassetta: il giorno che si desiderera' riutilizzarlo, potra' venire ricaricato da Cassetta e, quindi, venire eventualmente ricorretto o ristampato.

Non e'poco, vero?

Vi anticipiamo, inoltre, che, probabilmente, vi saranno delle implementazioni al programma che verranno pubblicate sui prossimo numeri di Pocket Pet.
Anche per questa ragione vi consigliamo di mantenere la numerazione di 100 in 100, come ve la proponiamo noi in modo da disporre dello spazio necessario per le implementazioni.

USO DEL PROGRAMMA

Una volta battuto e corretto il programma, potete dare il fatidico comando di RUN: vedrete apparire sullo schermo, in alto, la scritta:

- SONO DISPONIBILI .. N.. PAGINE

dove N e'il numero di pagine massimo che il Computer puo' memorizzare in funzione della propria memoria RAM libera.

E' intuitivo che, in presenza di eventuali programmi in Linguaggio Macchina, quali il Basic Plus, etc., vi sara' una minore disponibilita' di pagine.

Dopo qualche secondo, apparira' la pagina con la sua intestazione.

Come si vedra', le prime due righe sono necessariamente dedicate al display del tipo di ordini che si vorranno dare al Computer.

Il cursore lampeggera' all'inizio delle 23 righe disponibili, e si comportera' esattamente come fosse al di fuori di un programma; cioe' sara' movibile tramite i comandi usuali di EDIT di schermo, compresi lo SPACE e il DELETE.

A questo punto, si puo' scrivere il messaggio desiderato, o eseguire il disegno che si preferisce.

In caso di errore, le correzioni avvengono tramite i soliti tasti DELETE, CURSOR UP, CURSOR DOWN, anche shiftati.

Una volta soddisfatti del testo, lo si puo' memorizzare o stampare.

Noi vi consigliamo di memorizzare ogni testo, prima di compiere qualsiasi altra operazione: questo perche' qualsiasi errore possiate commettere, il testo si trovera' al sicuro. Inoltre, se per la stampa non e' necessario memorizzare, lo e' di certo per il passaggio su cassetta, che non avviene direttamente dalla memoria di schermo.

COME SI DANNO I COMANDI

Quando si desidera dare un comando, si deve premere il tasto "HOME CURSOR"; affinche'il cursore vada a lampeggiare nel riquadro piu' in alto e a sinistra dello schermo, cioe' al disopra della "X" dell'intestazione.
A questo punto si deve premere il comando "%" e si vedra' apparire la scritta in reverse "ATTENDO ORDINI".
Il PET e' in COMMAND MODE , cioe'attende che voi premiate uno dei seguenti tasti:

M- MEMORIZZA	L- CARICA DA CASSETTA
R- RICHIAMA	P- STAMPA LO SCHERMO
S- SALVA SU CASSETTA	C- CANCELLA LO SCHERMO

Se vi sbagliate e premete un tasto che non e' "%" , potete cancellare l'errore premendo SPACE o DELETE, e l'intestazione verra'riscritta in modo corretto.

Vediamo ora i comandi ad uno ad uno:

M - memorizza

Questo comando, una volta premuto, fara' apparire la scritta "NUM.PAGINA?" sull'intestazione.
Dovrete quindi decidere quale numero progressivo, da 0 al numero massimo di pagine consentito, potete dare alla vostra pagina.
Se userete un numero minore di 0 o maggiore del massimo consentito, apparira' la scritta "NON ESISTE".
Una volta battuto il numero, il cursore si spegnera', e il PET comincera' ad analizzare lo schermo, incasellando nella memoria libera successiva allo spazio occupato dal programma, i valori numerici relativi alla memoria di schermo dei caratteri rappresentati.
Terminata questa operazione, il cursore tornera' a lampeggiare all'inizio di pagina, e potrete riempire una nuova pagina di schermo.

R- richiama

Se desiderate rivedere cio' che avete memorizzato non avete che da premere "HOME CORSOR" , la "%" , e "R".
Vi apparira' di nuovo la domanda "NUMERO PAGINA".
Impostando il numero con cui avete codificato la pagina desiderata, vedrete, riga per riga, riapparire la vostra paginetta.
Semplice, vero?

P - stampa

Questo comando, fa si' che il PET analizzi lo schermo con lo stesso procedimento di "M", ma i dati rilevati, vengono tradotti da CODICE SCHERMO, in CODICE ASCII , e spediti alla stampante, che ricopia, percio', lo schermo, riga per riga.
Ricordate, quindi, che per stampare e' necessario richiamare sullo schermo la pagina desiderata, prima di premere "P".
Chi volesse potra', addirittura, modificare il programa in modo da far stampare direttamente dalla memoria. Questo potrebbe essere utile per testi molto lunghi, ma in questo caso, vi consigliamo di controllare che il testo non venga stampato sulla tratteggiatura della carta. Anche in questo caso, una volta terminato il lavoro, il cursore tornera'a lampeggiare, e potrete passare all'operazione successiva.

C - cancella

E un semplice comando di utilita', che permette di cancellare lo schermo.

S - salva su cassetta

Poiche'questo comando si riferisce alla memoria del PET, l'intestazione sparisce. Appaiono al suo posto delle domande:

- TITOLO TESTO ? (intesta il "file")

- FINO A QUALE PAGINA ? (da pagina 0 fino alla pagina ?)

Questo vi permette di creare un "file" contenente il testo da voi memorizzato, che potrete poi riutilizzare collocandolo nella posizione di memoria che piu' vi aggradera'.
Il processo e' un po' lungo, dato che il registratore si ferma ogni 3-4 secondi.
Abbiamo calcolato un tempo di circa due minuti per pagina. Pero' questo processo, non necessita di alcuna presenza umana, percio', puo'venire eseguito nei tempi morti.
Vi anticipiamo che stiamo studiando un metodo per velocizzare questa funzione.

L - carica da cassetta

Questo comando, come gia' detto, permette di caricare un "file" che avete gia' creato, collocandolo nel punto della memoria che desiderate.
Se percio', ad esempio, voi voleste aggiungere delle pagine a quelle memorizzate precedentemente su cassetta, potrete impostare la partenza della collocazione, dalla pagina 3, e avrete le pagine 0, 1, 2 libere per le vostre aggiunte.
Al termine del lavoro, potrete registrare di nuovo tutto su cassetta, e avrete il vostro testo nell'ordine sequenziale desiderato.
All'atto pratico, vi apparira' scritto:

- TITOLO DEL TESTO ?

- DA QUALE PAGINA ? (vi permette di collocare la partenza)

- FINO A QUALE PAGINA ? (collocazione fine del "file")

Anche questa funzione risente della limitazione temporale dell'uso della cassetta.

Ci sembra, a questo punto, di aver terminato la descrizione dell'uso del nostro programmino.
Sperando di essere stati esaurienti, passiamo all'analisi passo passo del programma.

REMarks

1000-1010	Viene testata la memoria disponibile e, sottratto lo spazio necessario per il programma, si computa il numero di pagine disponibili.
1200-1600	Stampa l'intestazione ed, eventualmente, il numero della pagina richiamata.
1700	Abilita il lampeggio del cursore durante il run del programma.
1800	Accetta il carattere da stampare.
1900	Non permette al cursore di spostarsi quando e' acceso, per evitare di lasciare reversato il carattere precedente.
2000-2200	Protezioni dell'intestazione (premendo SPACE l'intestazione viene riscritta).
2300	Testa la presenza del comando "%" alla locazione di schermo 32768.
2400-3500	Controllo tasti durante il "COMMAND MODE".
3600-4700	Ciclo di memorizzazione della pagina nella memoria libera.
4900-6100	Ciclo di richiamo pagine memorizzate.
6300-6800	Ciclo di SAVE su cassetta.
7000-7600	Ciclo di LOAD su cassetta.
7900-8300	Ciclo di stampa comprendente la formazione di una stringa di 40 car.
8400-8500	Espressione logica Booleana, che traduce il codice di schermo del PET, in codice ASCII.

EDITOR MAX

```
1000 PM=INT((FRE(0)-1000)/920):P=0:M=0
1100 PRINT"⌐SONO DISPONIBILI"PM"PAGINE":FORI=1TO2000:NEXT
1200 POKE167,1:PRINT"⌐"
1300 POKE216,0:POKE198,0:SYS57949:PRINT"x";:FORI=1TO29:PRINT" ";:NEXT:PRINT
1400 IFM=0THEN1600
1500 POKE216,0:POKE198,32:SYS57949:PRINT"PAG."P                              "
1600 FORI=1TO40:POKE32807+I,99:NEXT:PRINT
1700 POKE167,0
1800 GETA$:IFA$=""THEN1800
1900 IFPEEK(170)=1THEN1900
2000 PRINTA$;:IFPEEK(216)>23THENPOKE216,23
2100 IFPEEK(216)<2ANDPEEK(151)=6THENGOTO1300
2200 IFPEEK(216)<2ANDPEEK(151)=65THENGOTO1300
2300 IFPEEK(32768)=0THEN2500
2400 GOTO1700
2500 POKE216,0:POKE198,12:SYS57949:PRINT"■ ATTENDO ORDINI■";
2600 POKE32768,32:POKE32769,32
2700 GETA$:IFA$=""THEN2700
2800 FORI=0TO39:POKE32768+I,32:NEXT
2900 IFA$="M"THEN3600:REM MEMORIZZA
3000 IFA$="R"THEN4900:REM RICHIAMA
3100 IFA$="S"THEN6200:REM SAVE
3200 IFA$="L"THEN6900:REM LOAD
3300 IFA$="P"THENGOSUB7900:GOTO1200:REM STAMPA
3400 IFA$="C"THENM=0:GOTO1200REM CANCELLA
3500 GOTO2500
```

```
3600 POKE216,0:POKE198,5:SYS57949:PRINT"▓MEMORIZZA■";
3700 POKE216,0:POKE198,21:SYS57949:PRINT"                              "
3800 POKE216,0:POKE198,21:SYS57949:INPUT"NUM.PAGINA";P
3900 POKE216,0:POKE198,21:SYS57949:PRINT"                              "
4000 IFP<PMTHEN4500
4100 POKE216,0:POKE198,21:SYS57949:PRINT"NON ESISTE"
4200 FORI=1TO1000:NEXT
4300 POKE216,0:POKE198,21:SYS57949:PRINT"                              "
4400 GOTO3800
4500 POKE216,0:POKE198,27:SYS57949:PRINT"MEM.PAG."P
4600 FORI=0TO919
4700 POKE3000+(P*920)+I,PEEK(32848+I):NEXT
4800 M=0:GOTO1200
4900 POKE216,0:POKE198,5:SYS57949:PRINT"▓RICHIAMA■";
5000 POKE216,0:POKE198,21:SYS57949:PRINT"                              "
5100 POKE216,0:POKE198,21:SYS57949:INPUT"NUM.PAGINA";P
5200 POKE216,0:POKE198,21:SYS57949:PRINT"                              "
5300 IFP<PMTHEN5700
5400 POKE216,0:POKE198,21:SYS57949:PRINT"NON ESISTE"
5500 FORI=1TO1000:NEXT
5600 POKE216,0:POKE198,21:SYS57949:PRINT"                              ":GOTO5100
5700 POKE216,0:POKE198,27:SYS57949:PRINT"RIC.PAG."P
5800 FORI=0TO919
5900 POKE32848+I,PEEK(3000+(P*920)+I):NEXT
6000 POKE216,0:POKE198,28:SYS57949:PRINT"                      "
6100 M=1:GOTO1300
6200 GOSUB7700
6300 OPEN1,1,1,T$
6400 FORP=0TOK
6500 FORI=0TO919
6600 PRINT#1,PEEK(3000+(P*920)+I)
6700 NEXT:NEXT
6800 CLOSE1:M=0:GOTO1200
6900 GOSUB7700:INPUT"DA QUALE PAGINA ";N
7000 OPEN1,1,0,T$
7100 FORP=NTOK
7200 FORI=0TO919
7300 INPUT#1,A
7400 POKE3000+(P*920)+I,A
7500 NEXT:NEXT
7600 CLOSE1:GOTO1200
7700 INPUT"▓▓▓TITOLO TESTO:";T$
7800 INPUT"▓FINO A QUALE PAGINA";K:RETURN
7900 N=0:XT$="":OPEN4,4
8000 FORI=32768TO33767
8100 GOSUB8400:XT$=XT$+X$
8200 N=N+1:IFN>39THENGOSUB8500:N=0
8300 NEXT:CLOSE4:RETURN
8400 X=PEEK(I):X=(XAND127)OR((XAND64)*2)OR((64-XAND32)*2):X$=CHR$(X):RETURN
8500 PRINT#4,XT$:XT$="":RETURN
```

--*H*-----*H*-----*H*--

Nota del direttore responsabile:

Dato il considerevole valore di questo programma e la versatilita' di espansione fra tutti coloro che invieranno delle modifiche od aggiunte di particolare interesse all'EDITOR MAX verranno scelti i migliori dieci ai quali sara' inviata una cassetta contenente 10 programmi di giochi compatibili per vecchie e nuove ROMs.

ALTA RISOLUZIONE

F o r m a t a z i o n e

d i

c a r a t t e r i

Roberto Odoardi

Questo programma puo' essere considerato quale seguito dell'articolo apparso sul Pocket PET numero 0 che trattava dell'alta risoluzione ottenibile con le stampanti 2022/3022 e 4022.
Il primo risultato del programma e' proprio quello di mostrare sul video un reticolo da 7x6 necessario per la formatazione del carattere grafico desiderato.
Introducendo i riferimenti con delle coordinate vengono visualizzate, accendendosi o spegnendosi, le caselle interessate.
In caso di input non regolare, verranno visualizzati due asterischi (**); se i medesimi due asterischi verranno introdotti al posto delle coordinate vorra' dire che saremo arrivati al termine dell'introduzione.
Se invece si sostituiranno detti asterischi con altre coordinate potremo nuovamente proseguire con la formatazione del carattere speciale.
Premendo -RETURN- con i due asterischi si conclude l'introduzione dati, e si passa alla scelta del modo di stampare.

I comandi di stampa sono:

> + = stampa aggiungendo senza andare a capo;
> / = stampa con Line Feed.

Per terminare l'esecuzione del programma introdurre F anziche' le coordinate, cio' ci permettera' di ripristinare l'interlinea della stampante al valore normale di 24.

REMarks

- 100-160 Intestazione grafica.
- 170-240 Stampa del reticolo.
- 250-260 Input coordinate.
- 270-330 Test verifica carattere: coordinata della griglia, "**" oppure "F". Se A$ non e' compreso in questo campo il PET torna all'input.
- 340-360 Determina la posizione sullo schermo delle coordinate richieste e la varia da acceso in spento e viceversa.
- 370-410 Calcola mediante l'istruzione PEEK se ogni cellula del reticolo e' accesa o spenta e costruisce la stringa A$ contenente il carattere formatato.
- 420-430 Visualizza i numeri dei caratteri corrispondenti ai numeri binari rappresentati nella griglia.
- 440-470 Mostra i due simboli che indicano il tipo di stampa usato: "+" per stampare senza LF, "/" per stampare con LF ed attende che sia premuto uno dei due tasti.
- 480-520 Fase di stampa e ritorno all'inizio del programma.
- 530-540 Finale del programma con reset dell'interlinea dal valore usato nel programma (18) al valore standard (24).

```
100 REM ********************************
110 REM *** FORMATAZIONE CARATTERI ***
120 REM ********************************
130 REM *** DI  ROBERTO ODOARDI     ***
140 REM ********************************
150 REM *** POCKET GROUP  31 07 81 ***
160 REM ********************************
170 PRINT"  FORMATAZIONE 3022 4022  CHR N.";LL
180 PRINT"     A  B  C  D  E  F "
190 PRINT"    ┌";:FORL=1TO5:PRINT"──┬";:NEXT:PRINT"──┐"
200 FORI=1TO7
210 PRINTI;:FORL=1TO6:PRINT"│  ";:NEXT:PRINT"│"
220 PRINT"    ├";:FORL=1TO5:PRINT"──┼";:NEXT:PRINT"──┤"
230 NEXT
240 PRINT"    └";:FORL=1TO5:PRINT"──┴";:NEXT:PRINT"──┘     "
250 PRINT"                    **"
260 INPUT" QUALE COORDIN. ";A$
270 IFA$="**"THEN370
280 IFA$="F"THEN530
290 IFLEN(A$)<>2THEN330
300 IFLEFT$(A$,1)<"A"ORLEFT$(A$,1)>"F"THEN330
310 IFRIGHT$(A$,1)<"1"ORRIGHT$(A$,1)>"7"THEN330
320 GOTO340
330 PRINT"                    **                    ":GOTO260
340 G=32849+80*VAL(RIGHT$(A$,1))+(ASC(A$)-64)*3
350 IFPEEK(G)=32THENPOKEG,225:POKEG+1,97:GOTO260
360 POKEG,32:POKEG+1,32:GOTO260
370 A$="":FORI=0TO15STEP3:E=0:F=6
380 FORI1=559TO0STEP-80
390 G=PEEK(33492-I1+I)
400 IFG<>32THENE=E+2↑(F)
410 F=F-1:NEXT:A$=A$+CHR$(E):NEXT
420 PRINT"        ";:FORI=1TO6
430 PRINTRIGHT$("   "+STR$(ASC(MID$(A$,I,1))),3);:NEXT:PRINT
440 PRINT"             ";
450 PRINTTAB(27)" + APPEND"
460 PRINTTAB(27)"   /  "CHR$(34)" E LF"
470 GETT$:IFT$<>"+"ANDT$<>"/"THEN470
480 OPEN5,4,5:OPEN6,4,6:OPEN3,4:PRINT#6,CHR$(18)
490 PRINT#5,A$:PRINT#3,TAB(LL)CHR$(254)CHR$(141);
500 IFT$="/"THENPRINT#3:LL=LL+1
510 CLOSE3:CLOSE6:CLOSE5
520 GOTO170
530 OPEN6,4,6:PRINT#6,CHR$(24):CLOSE6
540 END
```

--*H*-----*H*-----*H*--

Per chi ha l' 8000

Se avete gia' provato ad eseguire PRINT CHR$(142), avrete notato che lo schermo dell'8032 si schiaccia e tutto si trasforma in maiuscolo e grafico. Per evitare la noiosa digitazione del comando si puo' eseguire la seguente azione che in un colpo solo ottiene il medesimo effetto:
Premere contemporaneamente i tre tasti -SHIFT- -SHIFT- e -"-.
Per ritornare alla condizione normale non esiste una azione sintetica, ma si dovra' eseguire: PRINT CHR$(14).

Elaborato da Paolo Zangrandi con la collaborazione del maestro Isidoro Cusberti.

POKE note

Valori corretti delle note per PET

Quelli che seguono sono i valori da caricare col comando POKE 59464 per ottenere le scale musicali con un corretto rapporto delle altezze. Va pero' notato che:

1 - il valore caricato con la POKE 59466 determina il timbro del suono, ma tale timbro non e' costante al variare dell'altezza; tuttavia esistono timbri che sono abbastanza costanti da non fornire variazioni significative per l'orecchio.

2 -la terza ottava, data la sensibile variazione in seguito al variante anche di una sola unita' del numero, non ha tutti i rapporti perfettamente corretti; d'altra parte,nella pratica e' raramente necessario utilizzare anche questa ottava.

1a OTTAVA

N.	Nota	Valore
1	SI	249
2	DO	233
3	do	221
4	RE	207
5	re	195
6	MI	187
7	FA	176
8	fa	164
9	SOL	156
10	sol	146
11	LA	138
12	la	130

2a OTTAVA

N.	Nota	Valore
13	SI	123
14	DO	117
15	do	109
16	RE	104
17	re	97
18	MI	92
19	FA	87
20	fa	81
21	SOL	77
22	sol	72
23	LA	68
24	la	64

3a OTTAVA

N.	Nota	Valore
25	SI	61
26	DO	57
27	do	54
28	RE	51
29	re	48
30	MI	45
31	FA	42
32	fa	40
33	SOL	38
34	sol	35
35	LA	33
36	la	31

DURATA

Durata	Valore
1/64	1
1/32	2
1/16	4
1/ 8	8
1/ 4	16
1/ 2	32
1	64

l'istruzione DEF FNx(X)

di Bruno Brazzoduro (i2WBB)

Il linguaggio Basic (Beginner All-purpose Symbolic Instruction Code), che in parole nostrane significa Istruzioni Simboliche Codificate Multiuso per Principianti, ovvero (ISCMP).

Certamente risulta alquanto contorta come definizione contratta, quindi lasciamola pure nella versione originale.

Il BASIC offre, a chi e' addetto piu' o meno ai lavori, diverse e svariate possibilita' di personalizzare i propri programmi.

Sopratutto conoscendone i vari "statements", fra i quali l'istruzione oggetto del presente sproloquio, di ridurre notevolmente la fatica di cercare modi contorti ed astrusi per fare cose piuttosto semplici.

L'istruzione DEF FNx(X) permette di definire a priori in un punto qualunque del programma (pero' generalmente e' buona norma farlo nelle prime linee dopo le eventuali DIM) delle funzioni ad una variabile, dove "x" puo' essere una qualunque delle 26 lettere dell'alfabeto, si hanno quindi a disposizione ben 26 funzioni diverse.

Le funzioni FNx devono avere, in sede di definizione, un argomento fittizio che permette di indicare le modalita' di come verra' trattata la variabile; nel contesto del programma possono avere un argomento qualunque, ad esempio un'espressione anche contenente altre funzioni.

Bisogna pero, per amor di precisione dire che una funzione FNx NON puo' contenere nel suo argomento la funzione stessa ne' funzioni che la utilizzino nella propria definizione.

Il formato delle definizioni e' il seguente:

```
numero riga DEF FNx(argomento)=espressione
```

Va' inoltre detto che il BASIC non permette di dare una definizione diretta di piu' piu' di una variabile in una funzione.

Elenchiamo di seguito alcuni esempi:

```
a)  100 DEF FNR(X)=INT(X*100+.5)/100

b)  110 DEF FND(X)=INT(X)+INT((X-INT(X))*100.5)/60
```

La prima istruzione (a) permette di eseguire ogni volta che lo si riterra' opportuno, l'arrotondamento del risultato desiderato a due cifre decimali.

```
200 INPUT"1.MO NUM.=";A
210 INPUT"2.DO NUM.=";B
220 B = A/B
230 PRINT B
240 B = FNR(B)
250 PRINT B
```

Se si assegna il valore di 1 ad A e 3 a B, il risultato che si ottiene al PRINT della linea 230 sara' "0.33333333", mentre dopo che la variabile e' stata manipolata dalla linea 240, il risultato sara' troncato e arrotondato a "0.33".

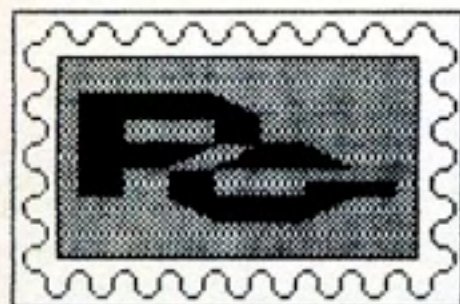
La seconda (b) permette la conversione del formato sessagesimale in decimale delle ore:minuti:secondi o gradi:minuti:secondi.

Nel trattamento delle frazioni di ore, per eseguire eventuali calcoli di "tempo x lire", e' senza dubbio necessario che queste siano congruenti. Cosa fare?: laboriose conversioni per troncare la parte intera, ricavare i minuti, e guarda caso, questi sono nel formato sessagesimale (sessantesimi di ore). Altra fatica per convertire 15 minuti affinche' rapportati al formato centesimale diano come risultato 0.25, il quale a questo punto puo' essere riaggiunto alle ore e quindi moltiplicato per le "svanziche".

Ebbene, usando la funzione (b) il tutto e' presto fatto in un colpo solo:

```
200 INPUT"ORE NEL FORMATO HH.MM";OM
210 INPUT"COSTO ORARIO";CO
220 CT = FND(OM) * CO
230 PRINT"LIRE =" CT
```

Se si assegna il valore di 4.15 ad OM e 1000 a CO il risultato CT dopo che la variabile e' stata trattata dalla funzione FND(X) sara' uguale a Lire 4250.

PROVARE PER CREDERE!

Il francobollo del Pocket Group

di

Gloriano Rossi e Luciano Odoardi

Gia' da quando il numero zero di Pocket PET e' andato in stampa esisteva un certo gruppo di persone, da me coordinate che esegue programmi, effettua prove e relazioni, modifica ed attualizza innumerevoli applicazioni.
Una prova evidente di tutto cio' e' proprio il Pocket PET ed anche i vari programmi che la Harden spa distribuisce tramite i propri fornitori.

Luciano, in un momento di pazzia, ha pensato di attribuire un marchietto al gruppo.
Cosi' e' nato il "francobollo del Pocket Group".
Per la mania che lo contraddistingue, Luciano si e' divertito ad utilizzare la sua stampante Commodore 3022 ed ha realizzato un programma che nella sua versione originale era composto di una cinquantina di righe, ma per non rubare troppo spazio sul Pocket PET ci ho messo le mani e.. ecco cosa e' saltato fuori:

```
110 REM *************************************************************
120 REM *****    F R A N C O B O L L O    P O C K E T    G R O U P    *****
130 REM *************************************************************
140 A$=CHR$(1):B$="╲":C$="╱":A$(1)=" ":FORI=1TO9:A$(1)=A$(1)+"⌒":NEXT
150 A$(2)=C$:FORI=1TO8:A$(2)=A$(2)+"◡":NEXT:A$(2)=A$(2)+B$
160 A$(3)=B$:FORI=1TO16:A$(3)=A$(3)+"▒":NEXT:A$(3)=A$(3)+C$
170 A$(4)=C$+MID$(A$(3),3,16)+B$:A$(5)=B$+MID$(A$(1),2,17)+RIGHT$(C$,1)
180 A$(6)=" "+MID$(A$(2),3,17)+RIGHT$(C$,1):B$="":C$=""
190 DIMC$(12):B$="┌":M$="‾":D$="┐":E$="|  ":F$="  |":G$="└":N$="_":I$="┘":O$=" "
200 FORI=1TO18:C$=C$+M$:H$=H$+N$:L$=L$+O$:NEXT
210 C$(1)=B$+C$+D$:C$(2)=E$+LEFT$(L$,16)+F$:C$(3)=E$+LEFT$(C$(1),15)+D$+F$
220 C$(4)=E$+LEFT$(C$(2),14)+F$+F$
230 C$(7)=G$+H$+I$:C$(6)=C$(2):C$(5)=E$+LEFT$(C$(7),15)+I$+F$
240 C$(8)=A$+"▚    ◥▀":C$(9)=A$+"▚ ▀   ▚ ▀":C$(10)=A$+"▚   ▀◤▚◤ ◥▀"
250 C$(11)=A$+"▚ ▀ ▚ ▀   ▬":C$(12)=A$+"   ◥▚   ▀◤"
260 OPEN1,4:PRINT#1:OPEN6,4,6:PRINT#6,CHR$(18):FORI=1TO4
270 PRINT#1,A$(I);CHR$(141);C$(I):NEXT:PRINT#6,CHR$(9):PRINT#1,TAB(3);C$(8)
280 FORI=0TO3:R=ABS(INT(I/2)-I/2)*2:PRINT#1,A$(R+3);CHR$(141);C$(4)
290 PRINT#1,TAB(3);C$(I+9):NEXT:PRINT#6,CHR$(9):PRINT#1,A$(3);CHR$(141);C$(4)
300 PRINT#6,CHR$(18):FORI=4TO6:PRINT#1,A$(I);CHR$(141);C$(I+1)
310 NEXT:PRINT#6,CHR$(24):CLOSE6:CLOSE1:GOTO260
```

Flussi RELative

di
Gloriano Rossi

Prima di iniziare a parlare direttamente dei files relative, vediamo quanti e quali tipi di flussi possono essere scritti e letti sulle unita' disco Commodore (2040-3040-4040-8050).

FLUSSO DI PROGRAMMA (tutte le versioni di BASIC e di DOS)

Contiene tutto il programma oggetto cosi' come si trova nell'unita' centrale (2001-3032-4032-8032) al momento della SAVE; il file contiene anche i vari puntatori specifici del programma stesso.

FLUSSO SEQUENZIALE (tutte le versioni di BASIC e di DOS)

Questo tipo di file e' il piu' classico dei flussi trattati in qualsiasi elaboratore.
Un record viene scritto uno dopo l'altro e parimenti un record viene letto uno dopo l'altro.
Tutti i records possono avere una propria lunghezza differente dalle lunghezze degli altri records del medesimo file e non esiste alcun identificatore di sistema relativo al singolo record.
Esiste esclusivamente un separatore (CHR$(13)) per identificare la fine del record e l'eventuale inizio del record successivo.

FLUSSO RANDOM (o accesso diretto) (DOS 1.0/2.x e BASIC 2/3/4)

La caratteristica principale di un flusso organizzato con il sistema RANDOM e' quella di poter accedere a qualsiasi record direttamente tramite l'uso di una chiave, senza dover essere costretti a leggere sequenzialmente il file e quindi testare ogni record fino a trovare quello voluto.
Un flusso organizzato con il sistema RANDOM e' costituito in realta' da due files.
Il primo file, quello piu' grosso e capiente, contiene tutti i records scritti sequenzialmente, cioe' uno dopo l'altro.
Il secondo file e' molto piu' piccolo ed il suo contenuto e' costituito dalle chiavi di ogni record del file principale con in piu' le

coordinate di posizione. Queste coordinate sono espresse in traccia, settore e posizione del byte di inizio record.
In sintesi viene riportato in questo file il numero del blocco ed il puntatore ove risiede o inizia il record interessato. Per queste ragioni un file organizzato in questa maniera e' chiamato piu' propriamente I.S., da Indexed Sequential poiche' i records vengono scritti sequenzialmente, ma si puo' accedere ad essi tramite indice.
A partire da questo tipo di organizzazione, il singolo record, o i campi relativi ad una informazione completa, devono essere di una lunghezza ben precisa.
Se noi definissimo un flusso anagrafico organizzato in maniera I.S. dovremo dire innanzitutto quanto e' lunga la chiave e quanto spazio occupa l'informazione e quanto grande (numero records stimato) dovra' essere il flusso.
Il vincolo di lunghezza e' dovuto principalmente alla necessita' di poter eseguire degli update (aggiornamenti) del file. Infatti se noi, ad esempio, non definissimo la lunghezza record potremo esclusivamente leggere o scrivere le informazioni, ma non potremo riscrivere un record dopo una correzione. In seguito di un update la lunghezza di un record potrebbe risultare piu' grande rispetto la versione precedente e quindi, al momento della riscrittura, si verificherebbe un accavallamento con il record successivo con conseguente perdita di questultimo.

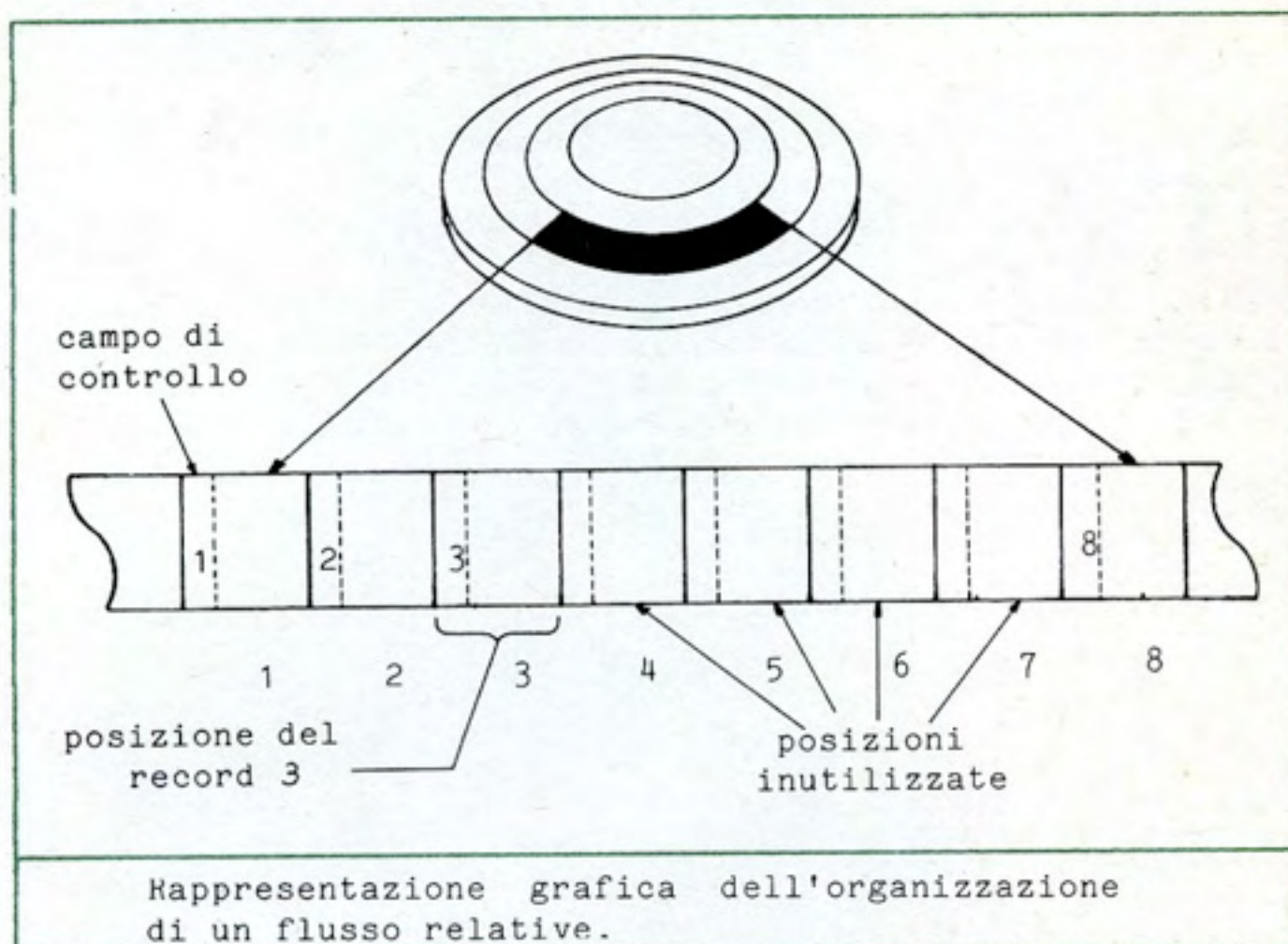
FLUSSO RELATIVE (DOS 2.x e BASIC 4)

Dopo aver capito o quantomeno aver assimilato i concetti base del flusso sequenziale e del flusso I.S. possiamo passare alla descrizione del flusso relative.
Questo tipo di organizzazione di un file e' forse la piu' intersessante rispetto a quelle precedenti in quanto offre maggiori vantaggi rispetto alle organizzazioni sequenziali e indexed, soprattutto in determinati tipi di applicazioni.
La differenza con un sequenziale sta nel fatto di poter accedere ai records sia sequenzialmente che in maniera diretta.
La differenza con un flusso RANDOM, sta nella mancanza del piccolo file delle chiavi e posizioni, nonche' per il tipo di scrittura che viene utilizzato.
Il flusso relative e' un file nel quale ogni record occupa una specifica posizione su disco.
Questo tipo di organizzazione consente all'utilizzatore di poter accedere direttamente ad ogni singolo record, senza dover essere costretti ad esaminarne altri (sequenziale) oppure senza dover effettuare ricerche preventive in un indice (RANDOM o I.S.).

Numero Relativo del RECORD.

L'accesso e la scrittura di ogni singolo record posto in un file del tipo relative sono effettuate tramite l'indicazione della posizione relativa del record.
Questa posizione e' calcolata a partire dall'inizio del flusso stesso.
La posizione relativa di un record prende il nome di "NUMERO RELATIVO"; questo numero non e' un indirizzo di disco, ma e' un valore intero e positivo che viene tradotto dal sistema nell'indirizzo del record che e' stato richiesto o che deve essere scritto o riscritto.

Il numero relativo di un record puo' essere parte del record stesso, oppure la risultante di determinate operazioni eseguite sul record o i dei campi che lo compongono.
Un sistema estremamente facile per determinare il numero relativo di un record e' quello di far coincidere questo numero con il numero chiave del record stesso.

Rappresentazione grafica dell'organizzazione di un flusso relative.

Se ad esempio, dobbiamo memorizzare un certo numero di articoli, il codice dell'articolo potrebbe corrispondere al numero relativo.
Cosi' l'articolo il cui codice corrisponde al numero 005 occupera' la quinta posizione nel file in oggetto.
Tutto questo ragionamento potrebbe essere valido per una ditta che possiede, ad esempio, mille articoli e li ha numerati progressivamente da 1 a 1000; il flusso da creare sara' costituito da 1000 porzioni di disco.
Generalmente pero' il numero di codice di un articolo ben difficilmente parte dal numero 1 e prosegue in avanti e a maggior ragione raramente i numeri di codice degli articoli sono assegnati sequenzialmente e non tutti i numeri vengono assegnati a relativi articoli.
Avremo ad esempio una ditta che produce 800 articoli i cui codici vanno dal numero 0001 al numero 2400.
Sara' illogico assegnare 2400 porzioni di disco per contenere quegli ottocento records, tre volte tanto il necessario.
Per questo caso, ed altri analoghi, si utilizza una tecnica detta di randomizzazione.

La Tecnica di RANDOMIZZAZIONE.

In pratica non si puo' dire che esiste una unica tecnica di randomizzazione.
Ogni sistema per individuare un numero relativo al record si puo' definire tecnica di randomizzazione.

Una tecnica ideale, per l'esempio riportato, e' quella di dividere il numero di codice per tre. Il valore intero risultante sara' il numero relativo di posizione del record nell'ambito del file.
Ogni tecnica di randomizzazione, pero', genera inequivocabilmente dei records sinonimi. vediamone un esempio:

num.codice		costante		risultato	posizione
260	:	3	=	86.6666	86
* 261	:	3	=	87	87
* 262	:	3	=	87.3333	87
* 263	:	3	=	87.6666	87
264	:	3	=	88	88

Che cosa e' successo?
I numeri di codice segnati con l'asterisco occuperebbero il medesimo posto nel file, cosa naturalmente impossibile.
Che cosa fare allora in questi casi?
Tutte le tecniche che si possono utilizzare in questi frangenti si riassumono in un unico termine:

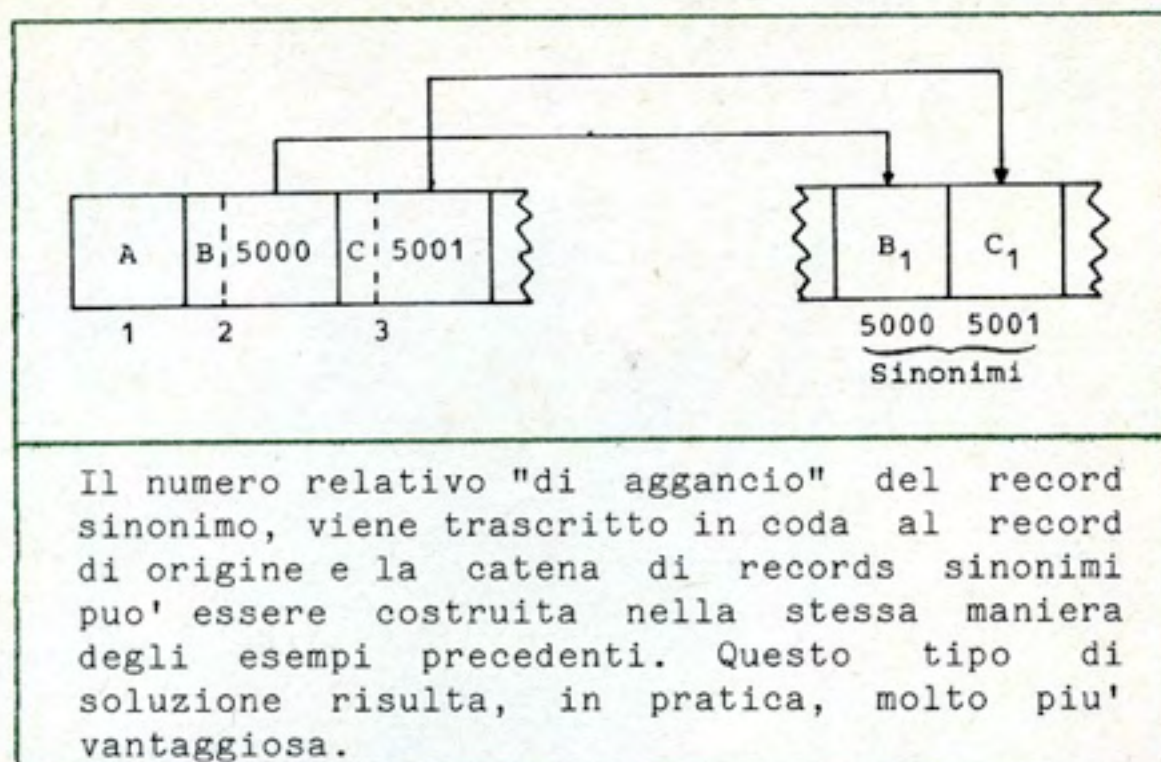
La Gestione dei SINONIMI.

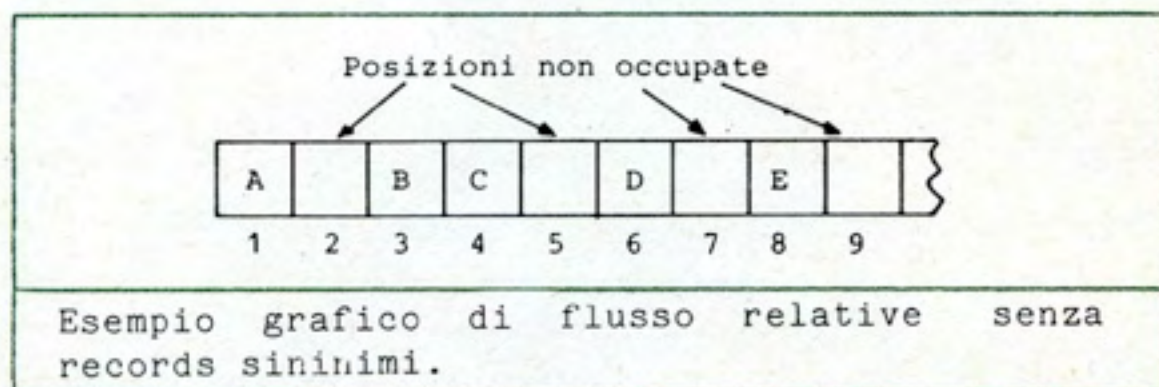
Il sistema piu' utilizzato in programmazione e' quello di dimensionare il file relative con uno spazio di circa il 15% maggiore del necessario (nel nostro caso 800 x 1.15 = 920, arrotondato per eccesso 999); quindi si definisce questa zona aggiuntiva quale "polmone" di riserva per i records sinonimi (da 801 a 999).

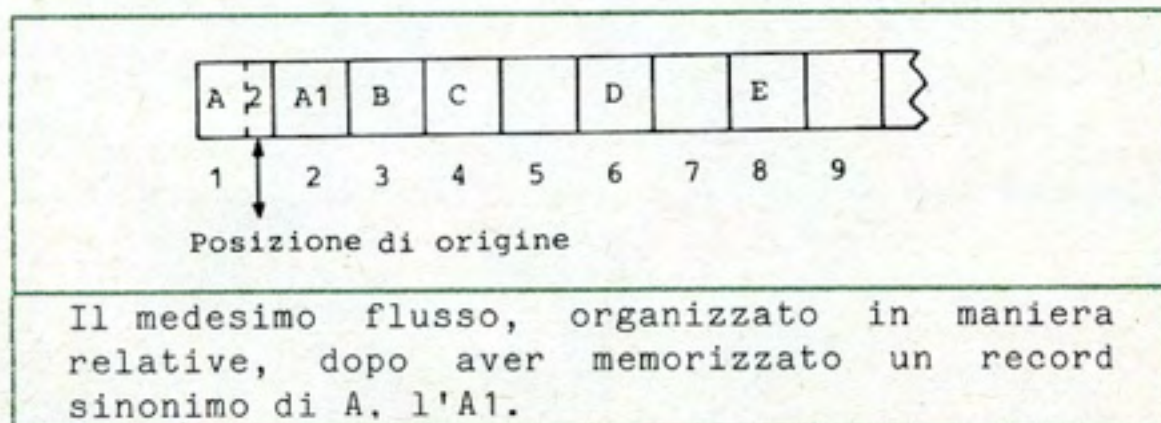
Il migliore metodo per la gestione dei sinonimi e' quello di riservare una certa zona del flusso.

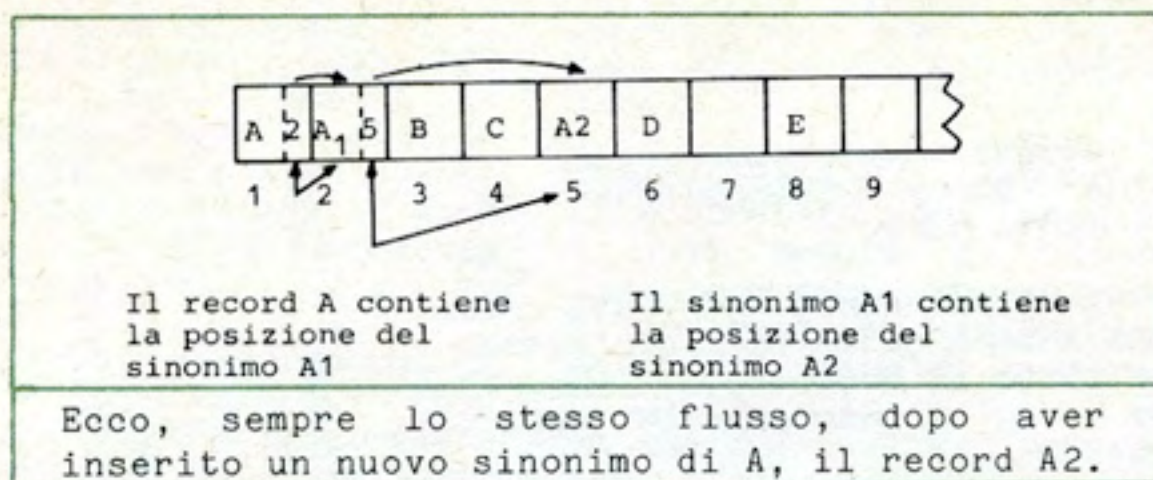
Se dobbiamo scrivere il record con codice 262 e ci troviamo la posizione 87 gia' occupata dall'articolo 261, dovremo riscrivere il record 87 con un campo in piu' che definira' la nuova chiave di collegamento (maggiore di 800).

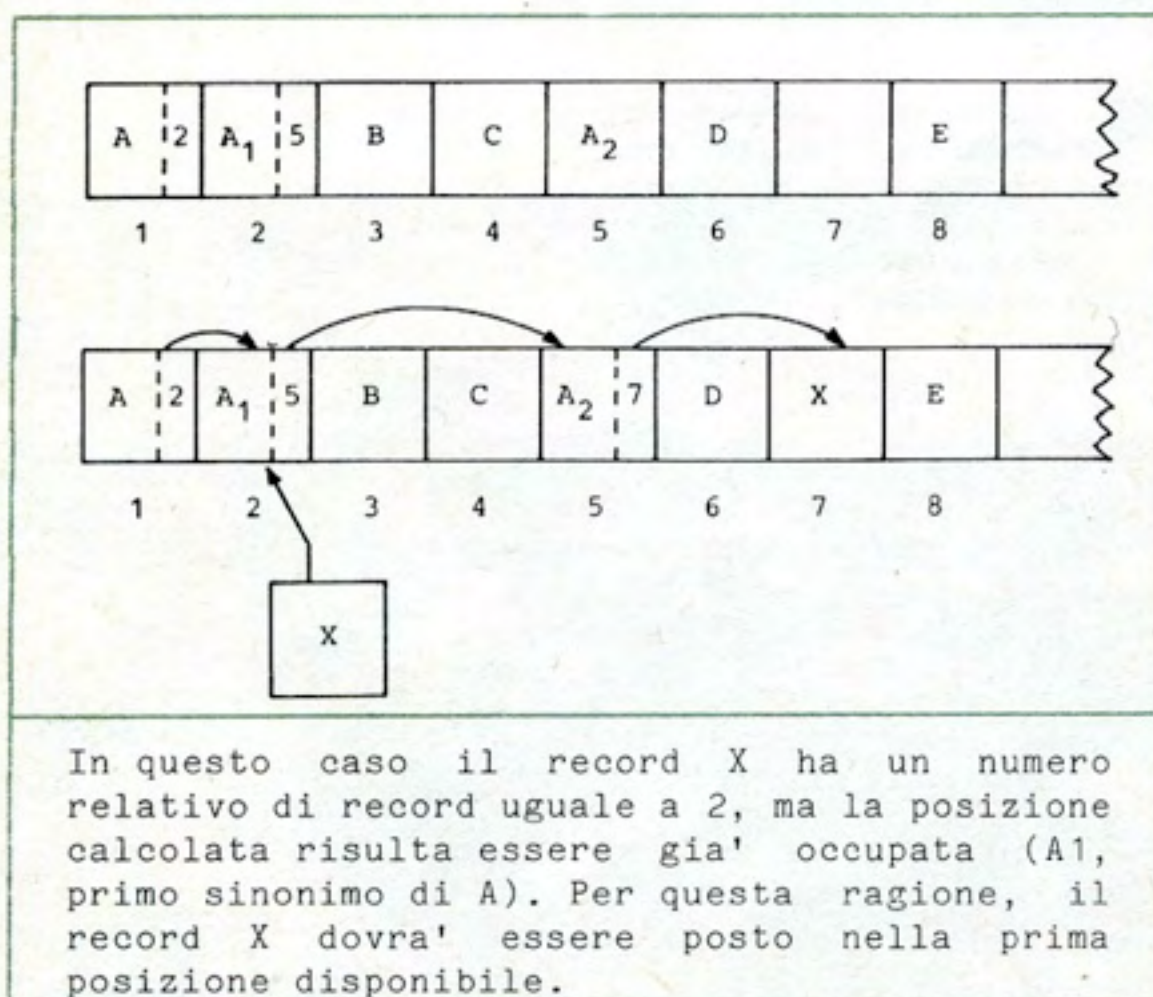
In fase di lettura si dovra' inevitabilmente testare se il record posto nella posizione calcolata sia effettivamente quello voluto.
In caso contrario si va ad individuare la chiave di collegamento che ci portera' in una nuova posizione del flusso.

Il numero relativo "di aggancio" del record sinonimo, viene trascritto in coda al record di origine e la catena di records sinonimi puo' essere costruita nella stessa maniera degli esempi precedenti. Questo tipo di soluzione risulta, in pratica, molto piu' vantaggiosa.

Per certi tipi di applicazione, pero', la zona dei sinonimi potrebbe risultare insufficente nonostante la valida capienza del file.
Un caso tipico con questo problema e' quello di una gestione di nomi e relativi numeri telefonici.
Il tipo di chiave che si vuole mantenere e' quello relativo al nome.
Potrebbe verificarsi che dopo la traduzione del nome in numero relativo si abbiano dei records sinonimi in numero statisticamente superiore alle previsioni.

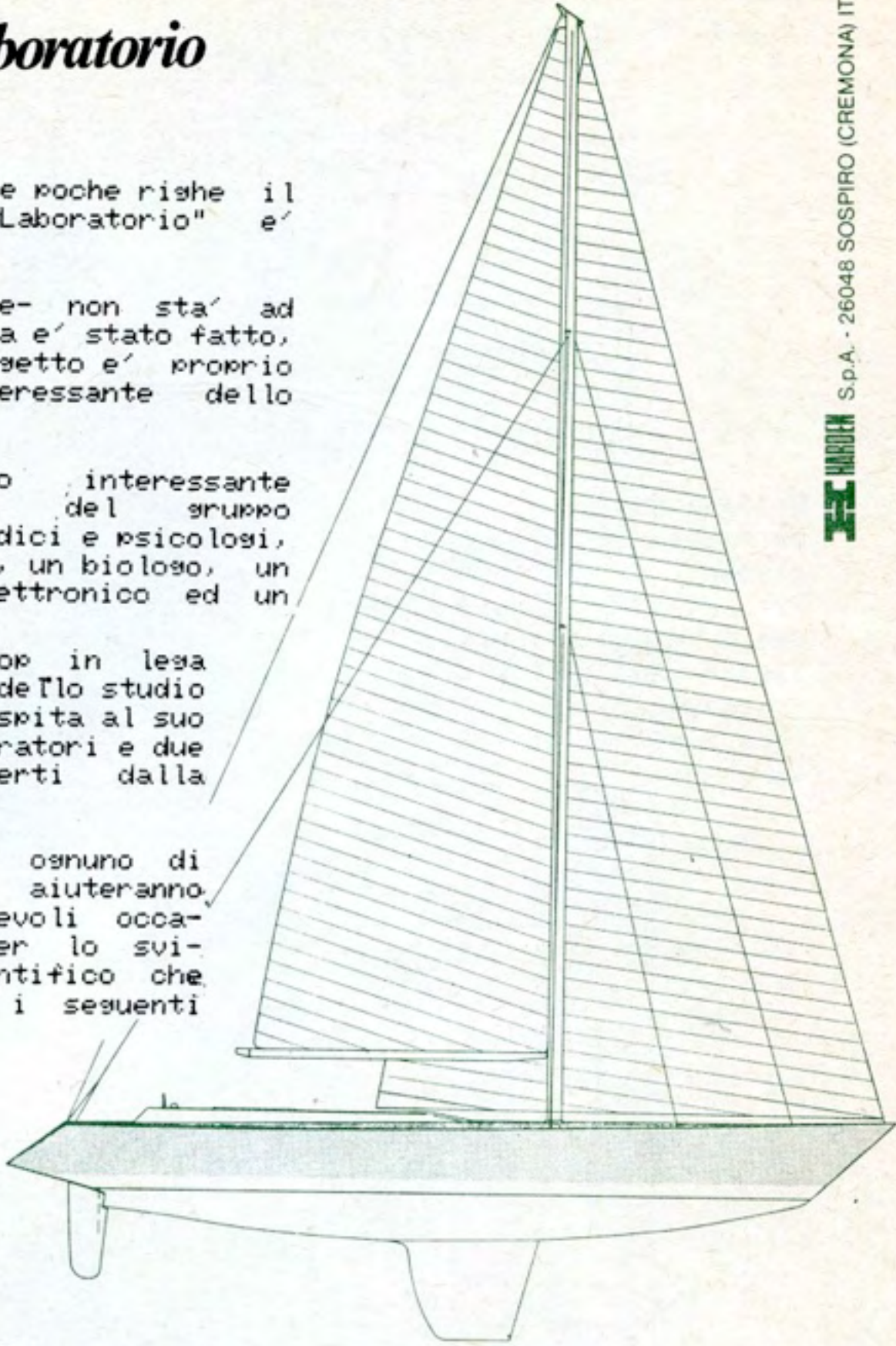
Esempio grafico di flusso relative senza records sininimi.

Si scrivono allora i records nelle rispettive posizioni calcolate, fino a quando non si trovi il posto occupato.

Il medesimo flusso, organizzato in maniera relative, dopo aver memorizzato un record sinonimo di A, l'A1.

Quando cio' avviene si ricerca la posizione libera piu' vicina, si esegue la registrazione. In fine si va a scrivere il numero di aggancio in coda al record precedente, il capo-catena.

Ecco, sempre lo stesso flusso, dopo aver inserito un nuovo sinonimo di A, il record A2.

Utilizzando questo sistema di gestione dei sinonimi, puo' accadere che un record debba occupare una posizione (primaria) occupata pero' da un elemento di una catena di sinonimi.
Per ovviare a questo incoveniente si va a ricercare l'ultimo elemento della catena "usurpante" e si registra il "malcapitato" nella posizione libera piu' vicina riportando sempre il numero relativo nel record precedente. Avremo trattato il nuovo record allo stesso pari di un record sinonimo della catena.

In questo caso il record X ha un numero relativo di record uguale a 2, ma la posizione calcolata risulta essere gia' occupata (A1, primo sinonimo di A). Per questa ragione, il record X dovra' essere posto nella prima posizione disponibile.

BREVE CONCLUSIONE

L'uso dei flussi relative visti in questa luce e' da considerarsi terreno valido per i soli programmatori esperti, senza con questo sminuire o demoralizzare gli iniziati. Con un po' di esperienza e numerose prove precedute da una analisi approffondita e' possibile eseguire una gestione sofisticata di un file relative.

La Barca Laboratorio

Mentre voi leggete queste poche righe il progetto "La Barca Laboratorio" e' senzaltro in alto mare.

Il termine -in alto mare- non sta' ad indicare che ancora nulla e' stato fatto, ma anzi che tutto il progetto e' proprio nella fase piu' interessante dello svolgimento.

Per preparare questo interessante progetto fanno parte del gruppo organizzatore, alcuni medici e psicologi, un fisico, un biochimico, un biologo, un geologo, un ingegnere elettronico ed un oceanografo.

La barca, che e' uno sloop in lega leggera, e' un progetto dello studio Giorgetti & Magrini ed ospita al suo interno due piccoli laboratori e due computer Commodore offerti dalla Harden spa.

I due PET/CBM, completi ognuno di unita disco e stampante, aiuteranno i navigatori in innumerevoli occasioni, ma soprattutto per lo sviluppo del programma scientifico che tocchera' principalmente i seguenti punti:

1 - Ricerche sull'uomo:
  - Ricerca biologica
  - Turni di lavoro
  - Sonno
  - Comportamento onirico
  - Dinamiche comportamentali
  - Aspetti dietetici e nutrizionali

2 - Ricerche oceanografiche:
  - Temperatura
  - Umidita'
  - Pressione
  - Vento
  - Radiazione solare
  - Studio generale sul moto ondoso
  - Studio generale dell'interfaccia aria-mare
  - Studio del movimento marino
  - Studio sull'inquinamento

3 - Ricerche tecnologiche:
  - Analisi strutturale delle imbarcazioni

4 - Esperimenti preliminari a terra ed in mare.

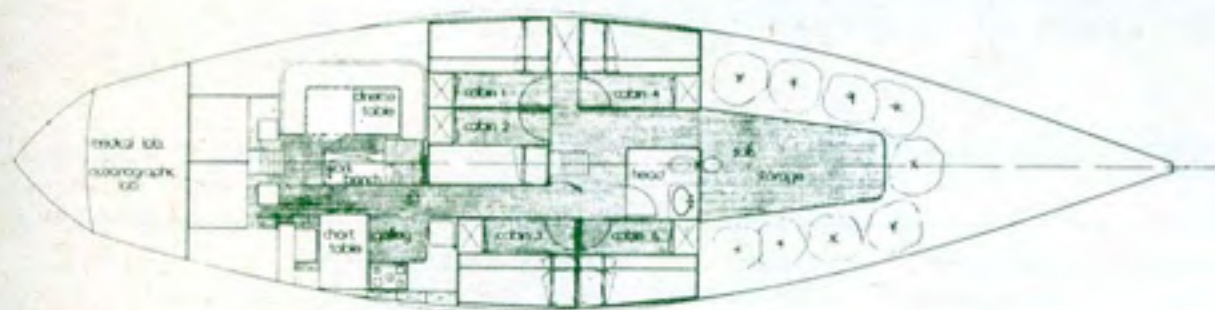
5 - Esperimenti con il laboratorio medico.
  - Esperimenti con il laboratorio oceanografico

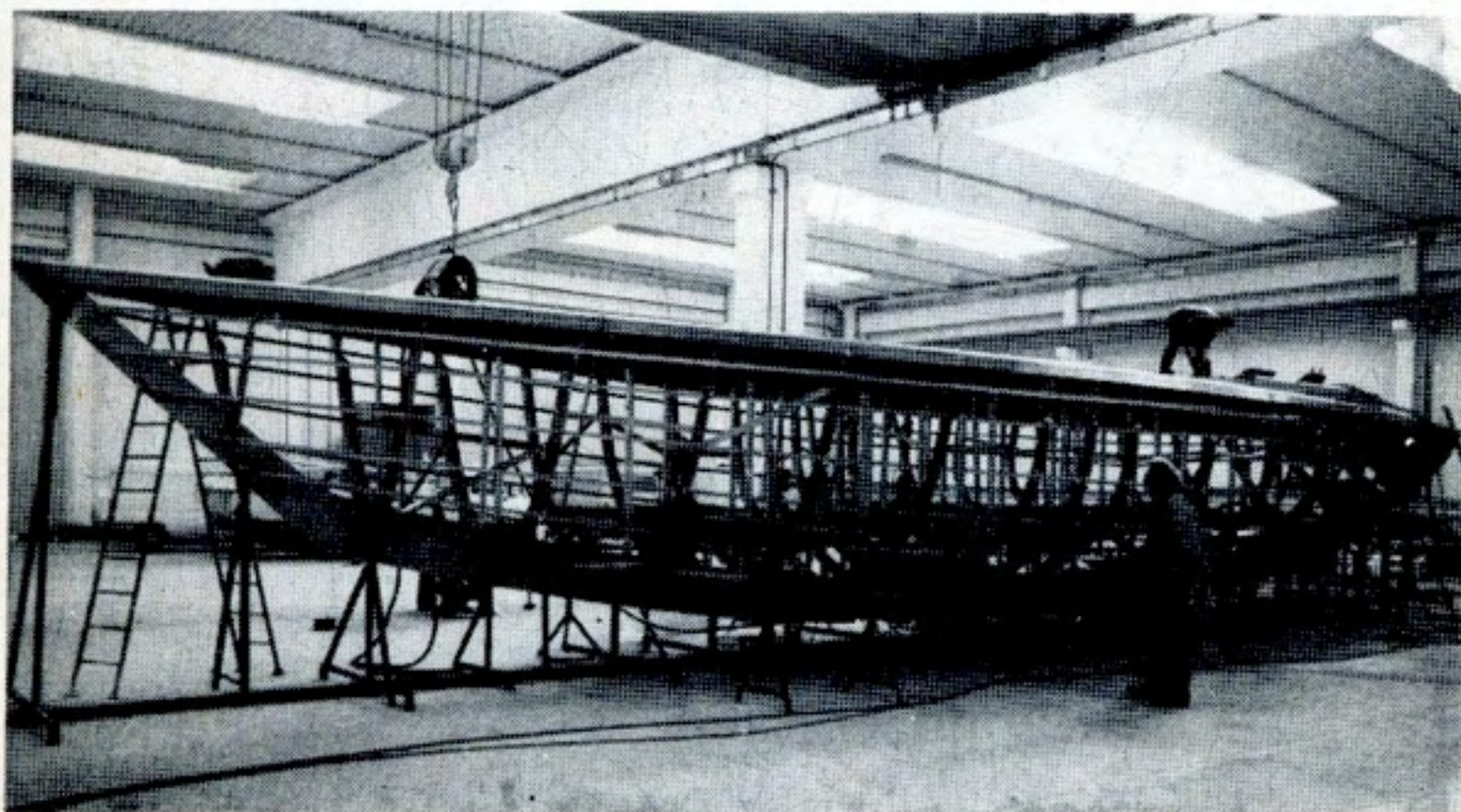
Tutti i dati forniti dalle sofisticate apparecchiature scientifiche e non, che trovano posto sulla barca laboratorio, sono poi analizzate dal 3032 che fornira' utilissimi elaborati e stampe statistiche.
Fra le varie applicazioni del PET sul monoalbero ci sara' quella di elaborare i dati relativi all'attivita' onirica in maniera da calcolare un giusto rapporto veglia-sonno al fine di ottenere un perfetto equilibrio psico-fisico.

Gia nel precedente Salone Nautico di Genova (18-27 ottobre 1980) il computer Commodore e' stato utilizzato per fornire spiegazioni sull'attivita de "La Barca Laboratorio" e del suo programma.

Al termine della regata, il PET, verra' utilizzato ancora per elaborare quei dati che durante la navigazione non e' possibile analizzare.

Questo -sposalizio- UOMO-MARE-SCIENZA e COMPUTER sara' seguito da tutti gli organi di stampa specializzata e non, nonche' dalle tre reti televisive nazionali e dalle emittenti private.

Un qualche cosa di grosso, dunque, con il nostro 'piccolo-potente' PET.

TECHNICAL DATA:

Overall length	19.41 m
Waterline length	15.70 m
Max. Width	5.16 m
Displacement	20.50 t
J	7.40 m
I	24.17 m
LPG	11.10 m
P	22.50 m
E	6.55 m
RSAT	197.68 m²

--*H*-----*H*-----*H*--

Una verita':

L'elaboratore e' incredibilmente veloce, preciso e ... stupido.
L'uomo e' straordinariamente lento, impreciso e ... creativo.
L'unione dei due costituisce una forza incalcolabile.

(Leo Cherne)

PAGINE > DI ZERO

Sul numero scorso di POCKET PET abbiamo riportato la pagina zero della memoria del PET-CBM con BASIC versione 3.
In questo numero riportaiamo quelle rimanenti pagine che hanno un particolare interesse per chi vuole utilizzare il linguaggio macchina per le sue elaborazioni.
Una prima parte comprende quelle pagine di memoria bassa che il microprocessore 6502 utilizza per funzioni "secondarie", quali ad esempio aree di buffers od altro.
Queste pagine sono esattamente la 1, la 2 e la 3.
In particolare e' da notare i due buffers delle tape cassette di cui il secondo normalmente viene utilizzato per routinette in linguaggio macchina dato che questo buffer, generalmente non viene utilizzato.
Veniamo ora alle seguenti pagine di memoria.
Possiamo notare che le pagine che vanno dalla 4 alla 128 sono proprio quelle che si utilizzano generalmente per i programmi BASIC.
Cio' che noi vediamo su video e' compreso fra le pagine 129 e 144, in

Pagina 1 - <256-511>

R O M s nuove	R O M s vecchie	semplice valore nuove	semplice valore vecchie	Descrizione
256-su'	256-su'	32	32	Area di lavoro lettura nastro (fino a 511) e memoria per conversione 256-318 per correzione errori lettura nastro (62 bytes) 256-266 conversione da binario-ASCII (11 bytes)
511-giu'	511-giu'	44	0	Stack (giu' fino a 256)

Pagine 2 e 3 - <512-1023>

R O M s nuove	R O M s vecchie	semplice valore nuove	semplice valore vecchie	Descrizione
512-592				buffer per riga input BASIC-80 bytes
		12597		512-513 0200-0201 contatore di progr.
		50		514 0202 status di processo
		0		515 0203 accumulatore
		171		516 0204 indice X
		0		517 0205 indice Y
		0		518 0206 puntatore stack
		15104		519-520 0207-0208 IRQ modific. utente
593-602		4		Tavola dei numeri logici dei files aperti
603-612		4		Tavola dei numeri delle periferiche relative ai files aperti
613-622		255		Tavola degli indirizzi secondari dei files aperti
633		28		Buffer della tastiera (10 bytes)
634-825		28		Buffer del tape cassette 1 -192 bytes
826-1017		173		Buffer del tape cassette 2 -192 bytes
1018-1019		59383		Vettore per il monitor in liguaggio macchina
1020-1023		195		Spazio per utilita' - non usato

particolare le prime quattro pagine costituiscono l'immagine vera e propria, mentre le seguenti vengono utilizzate dal sistema per la gestione dello schermo.

Se apriamo il PET, possiamo notare che la piastra dei componenti possiede tre posizioni per circuiti integrati grossi, inutilizzati. Questi spazi possono essere utilizzati per espansioni ROM, infatti la trasformazione da BASIC 3.0 a BASIC 4.0 prevede priprio di occupare una di queste tre zone libere.
Un altro utilizzo dell'espansione ROM potra' essere quella dell'uso del VisiCalc, che prevede proprio l'utilizzo di alcune routines site in quelle locazioni di memoria.

Per finire, riportiamo, le pagine che vanno dalla 193 alla 232, che contengono gli indirizzi dei puntatori delle routines del BASIC. Sarebbe stato inutile ripetere i vari indirizzi del BASIC e quindi terminano qui le pagine di memoria del PET-CBM con BASIC versione 3.0.

Sul prossimo numero riporteremo tutte le pagine dei PET-CBM con BASIC 4.0, cioe' i PET 4032 e 8032.

Pagine 4 - 128 - <1024-32767>

ROMs nuove	ROMs vecchie	semplice valore nuove	semplice valore vecchie	Descrizione
1024-32767		0		Area programmi utente ed espansione memoria RAM 4K PET: 1024-4095 0400-0FFF Area per programmi utente 4096-32767 1000-7FFF Espansione RAM 8K PET: 1024-8191 0400-1FFF Area per programmi utente 8192-32767 2000-7FFF Espansione RAM 16K PET:1024-16383 0400-3FFF Area per programmi utente 16384-32767 4000-7FFF Espansione RAM 32k PET:1024-32767 0400-7FFF Area per programmi utente

Pagine 129 - 144 - <32768-36863>

ROMs nuove	ROMs vecchie	semplice valore nuove	semplice valore vecchie	Descrizione
32768-36863	usuale	32	12	RAM VIDEO 32768-33767 visualizzazione memoria (1000 bytes)

Pagine 145 - 192 - <36864-49151>

ROMs nuove	ROMs vecchie	semplice valore nuove	semplice valore vecchie	Descrizione
36864-49151	usuale	144	0	Espansione ROM

Pagine 193 - 232 - <49152-59391>

ROMs nuove	vecchie	semplice valore nuove	vecchie	Descrizione
49152-49153	usuale	51008	50973	Puntatore - 1 a END (*)
49154-49155	usuale	50775	50760	Puntatore - 1 a FOR
49156-49157	usuale	52255	52277	Puntatore - 1 a NEXT
49158-49159	usuale	51199	51183	Puntatore - 1 a DATA
49160-49161	usuale	51878	51909	Puntatore - 1 a INPUT#
49162-49163	usuale	51904	51935	Puntatore - 1 a INPUT
49164-49165	usuale	53090	53104	Puntatore - 1 a DIM
49166-49167	usuale	51974	52003	Puntatore - 1 a READ
49168-49169	usuale	51372	51356	Puntatore - 1 a LET
49170-49171	usuale	51116	51100	Puntatore - 1 a GOTO
49172-49173	usuale	51076	51060	Puntatore - 1 a RUN
49174-49175	usuale	51247	51231	Puntatore - 1 a IF
49176-49177	usuale	50991	50956	Puntatore - 1 a RESTORE
49178-49179	usuale	51087	51071	Puntatore - 1 a GOSUB
49180-49181	usuale	51161	51145	Puntatore - 1 a RETURN
49182-49183	usuale	51266	51250	Puntatore - 1 a REM
49184-49185	usuale	51006	50971	Puntatore - 1 a STOP
49186-49187	usuale	51282	51266	Puntatore - 1 a ON
49188-49189	usuale	55055	55041	Puntatore - 1 a WAIT
49190-49191	usuale	65492	65492	Puntatore - 1 a LOAD
49192-49193	usuale	65495	65495	Puntatore - 1 a SAVE
49194-49195	usuale	65498	65498	Puntatore - 1 a VERIFY
49196-49197	usuale	53900	53908	Puntatore - 1 a DEF
49198-49199	usuale	55046	55032	Puntatore - 1 a POKE
49200-49201	usuale	51594	51582	Puntatore - 1 a PRINT#
49202-49203	usuale	51626	51614	Puntatore - 1 a PRINT
49204-49205	usuale	51050	51012	Puntatore - 1 a CONT
49206-49207	usuale	50612	50599	Puntatore - 1 a LIST
49208-49209	usuale	50550	51055	Puntatore - 1 a CLR
49210-49211	usuale	51600	51588	Puntatore - 1 a CMD
49212-49213	usuale	65501	65501	Puntatore - 1 a SYS
49214-49215	usuale	65471	65471	Puntatore - 1 a OPEN
49216-49217	usuale	65474	65474	Puntatore - 1 a CLOSE
49218-49219	usuale	51836	51870	Puntatore - 1 a GET
49220-49221	usuale	50522	50512	Puntatore - 1 a NEW
49222-49223	usuale	56133	56075	Puntatore - SGN (**)
49224-49225	usuale	56280	56222	Puntatore - INT
49226-49227	usuale	56164	56106	Puntatore - ABS
49228-49229	usuale	0	0	Puntatore - 1 a Puntatore USR
49230-49231	usuale	53849	53860	Puntatore - FRE
49232-49233	usuale	53882	53893	Puntatore - POS
49234-49235	usuale	56926	56868	Puntatore - SQR
49236-49237	usuale	57215	57157	Puntatore - RND
49238-49239	usuale	55542	55487	Puntatore - LOG
49240-49241	usuale	57050	56992	Puntatore - EXP
49242-49243	usuale	57304	57246	Puntatore - COS
49244-49245	usuale	57311	57253	Puntatore - SIN
49246-49247	usuale	57384	57326	Puntatore - TAN
49248-49249	usuale	57484	57416	Puntatore - ATN
49250-49251	usuale	55016	55014	Puntatore - PEEK
49252-49253	usuale	54870	54868	Puntatore - LEN
49254-49255	usuale	54079	54089	Puntatore - STR$
49256-49257	usuale	54919	54917	Puntatore - VAL
49258-49259	usuale	54885	54883	Puntatore a ASC
49260-49261	usuale	54726	54724	Puntatore a CHR$
49262-49263	usuale	54746	54744	Puntatore a LEFT$
49264-49265	usuale	54790	54788	Puntatore a RIGHT$
49266-49267	usuale	54801	54799	Puntatore a MID$
49268-49297				Gerarchia ed indirizzi di azione Per operatori
49298-49553				Tavola delle parole chiave BASIC
49554-49833				Messaggi d'errore BASIC

Cross Reference

di
Gloriano Rossi

Prima di parlare del programma CROSS REFERENCE, voglio spiegare, per chi non lo sapesse, che cosa e' e a cosa serve una lista tipo quella generata dal programma in oggetto.

Sui grossi elaboratori, dove si utilizzano linguaggi piu' evoluti tipo ad esempio il COBOL, quando si intorduce un nuovo programma per una determinata elaborazione, e' necessario eseguire in un primo tempo una compilazione e quindi un linker.

Durante la fase di compilazione il programma sorgente viene analizzato completamente al fine di individuare eventuali errori di sintassi o di incongruenza; quindi se tutto va bene, il compilatore, provvede a tradurre il programma in un linguaggio praticamente vicino al linguaggio macchina.

Il linker alfine provvede a sistemare questo gruppo di istruzioni in una libreria oggetto.

Fra i vari options che il compilatore offre al programmatore ce ne' uno di particolare interesse. Questo option e' costituito proprio da una mappa dove vengono in una prima parte elencate tutte le variabili con le relative locazioni dove esse vengono trattate, ed in una seconda ed ultima parte le Labels (etichette) e le relative posizioni dove esse vengono richiamate.

In un programma particolarmente breve l'uso del CROSS REFERENCE risulta pressoche' inutile, in quanto tutto e' sott'occhio e con una unica visuale si ha la situazione generale del programma in esame.

La faccenda diventa un po' "acida" quando questo programma incomincia ad avere delle dimensioni un po' piu' grandi. Una variabile si trova qui, poi nell'altro foglio, poi ancora nell'ultima pagina, ma ce ne era una anche da un'altra parte!, quindi segna qui, segna di la' e... dopo un po' e' un caos tale che non si capisce piu' niente.

Con il risultato del CROSS REFERENCE tutto diventa piu' semplice: si va a vedere la variabile e quindi, da quella mappa, si arriva direttamente nelle posizioni interessate senza dover diventare, si puo' ben dire, matti.

Oramai lo sanno tutti, il PET/CBM utilizza un interprete BASIC e non un compilatore. Questo fatto ci permette di evitare la noiosa attesa dell'esecuzione della compilazione, nonche' di avere sempre sottomano il programma sorgente/oggetto per un qualsiasi update od altra consultazione.

Tutto cio' e' a discapito pero' della funzione che e' stata descritta fino ad ora.

Per ovviare a questo fatto e' stato studiato un programma apposito che legge su disco le istruzioni del programma da esaminare ed individua quelle informazioni necessarie per completare la REFERENCE MAP.

Il programma CROSS REFERENCE e' stato realizzato in un primo tempo in una forma un po' scialba. Io ho provveduto a migliorarlo e modificarlo al fine di renderlo versatile ed utilizzabile nel migliore dei modi.

L'ultima versione, quella che vi propongo, vi permette di accedere a qualsiasi file/programma, scritto su disco, ed ottenerne sia una mappa per variabili che una mappa per righe.

Non esistono limitazioni nell'utilizzo del CROSS REFERENCE, ne' per la versione di BASIC del vostro PET ne' tantomeno dell'unita' disco utilizzata.

CROSS REFERENCE gira quindi sia sulla serie 2001 che sulla serie 3000, nonche' sulla serie 4000 e 8000. Allora qualsiasi sia la configurazione del vostro sistema si potra' utilizzare questo programma.

Per chi possedesse il PET/CBM 8032, consiglio di modificare la riga 770 nel modo seguente:

```
770 C=3:Z=12:IF Z$="S" THEN C=4
```

Si notera' che la modifica riguarda essenzialmente la variabile -Z-; il contenuto di questa variabile influenza il numero relativo delle righe riportate sulla medesima linea. Il computer modello CBM 8032, si sa, ha 80 colonne di video, come d'altronde 80 colonne sono utilizzate sulla stampante 3022 o 4022, quindi e' opportuno unificare il valore di "a capo linea" al fine di non occupare troppo spazio sullo schermo.

COME SI USA IL CROSS REFERENCE?

Si e' appena terminato un programma che chiameremo PRG-X e si vuole ottenerne sia la mappa delle variabili che quella delle linee.

Il primo passo da eseguire sara quello di salvare con una normale SAVE il PRG-X su dischetto. Il quale sara' situato nel drive zero.

Si carica a queso punto il programma CROSS REFERENCE e si impartisce il comando RUN.

Il CROSS REFERENCE chiedera' subito se si vuole una mappa per linee o per variabili; sara' sufficente per dare questa risposta battere la lettera -L- oppure la lettera -V-.

La seconda domanda che il CROSS vi rivolgera' sara' inerente al nome del programma da esaminare, e quindi voi batterete: "PRG-X".

A questo punto il CROSS andra' a ricercare sul disco inserito nel drive zero il programma richiesto e, se lo trova, incomincera' ad esaminarlo linea per linea fino alla fine.

Se il programma PRG-X risultasse eccessivamente grosso e' consigliabile modificare le dimensioni delle tabelle site in riga 250, in particolare quelle relative ad X$ e a C. La riga 250 potra', limitatamente alla memoria del vostro PET, essere modificata in questo modo:

```
250 DIM A$(15),B$(3),X$(800),C(400)
```

Il risultato finale potra' essere visualizzato sul video (C=3 sulla riga 770) oppure stampato su carta tramite la stampante di sistema (C=4 oppure C=5 sempre nella riga 770).

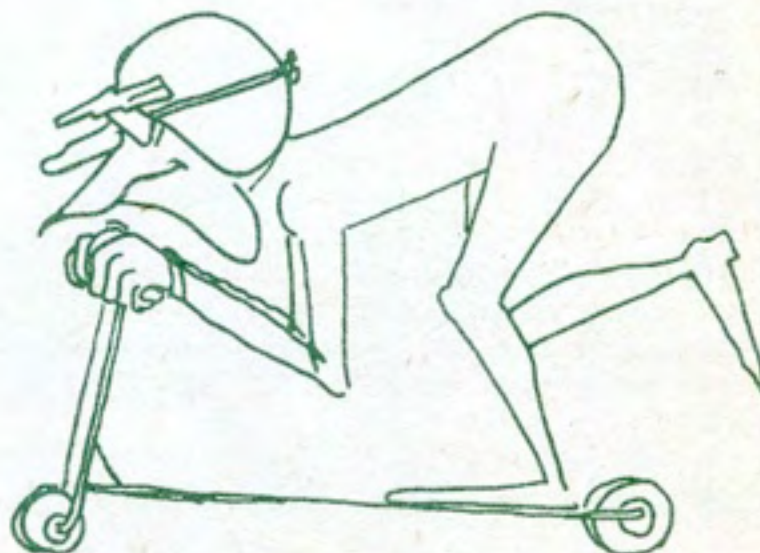
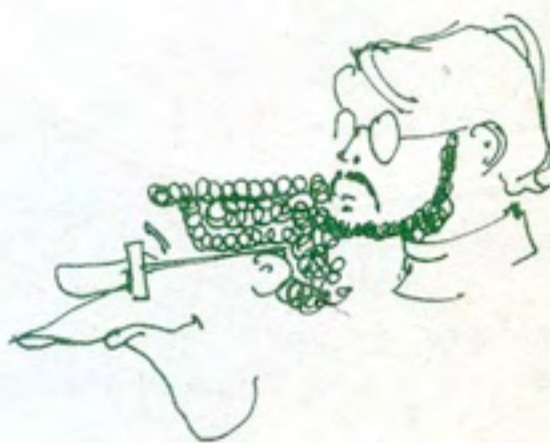
Per chi possedesse una stampante tipo LINA e' necessario modificare il numero di apertura del file di stampa che nel CROSS corrisponde a 2. Modificare quindi la OPEN 2,C di riga 780 in OPEN 130,C ;la CLOSE 2 di riga 900 in CLOSE 130 ed infine tutte le PRINT.2 in PRINT.130 (righe 780,790,800,810,860,880,890 e 900).

Il listato del programma CROSS REFERENCE e' seguito dalle due mappe, una per variabili e l'altra per linee, del CROSS REFERENCE stesso.

Noterete da soli, esaminando attentamente le due mappe, la validita' e l'utilita' di questo programma e... buone "mappate"!!

Cross Reference

```
100 REM**********************************************************************
110 REM***                          **                                    ***
120 REM***  C R O S S               **   P O C K E T     G R O U P        ***
130 REM***                          **                                    ***
140 REM***      R E F E R E N C E   **   GLORIANO ROSSI        I2KH       ***
150 REM***                          **                                    ***
160 REM**********************************************************************
170 REM
230 MA$="-------*------------------------- - - -"
240 MB$="VAR.  ! LINEA DEL PROGRAMMA "
250 DIM A$(15),B$(3),X$(500),C(255)
260 PRINT"" C R O S S - R E F E R E N C E"
270 Q$=CHR$(34):S$="      ":B$(1)=Q$:B$(3)=CHR$(58)
280 PRINT"LISTA VARIABILI A LINEE   V";:INPUTZ$
290 IFZ$="V"ORZ$="L"THEN310
300 PRINT"TTT":GOTO280
310 C2=5:IFZ$="L"THENC2=6
320 FORJ=1TO255:C(J)=4:NEXTJ:FORJ=48TO57:C(J)=6:NEXTJ
330 IFC2=5THENFORJ=65TO90:C(J)=5:NEXTJ:FORJ=36TO38:C(J)=7:NEXTJ:C(40)=8
340 C(34)=1:C(143)=2:C(131)=3
350 PRINT"NOME PROGRAMMA   *";:INPUTP$:IFP$="*"THENPRINT"TTT":GOTO350
360 IFP$="*"THENPRINT"TTT":GOTO350
370 OPEN1,8,3,"0:"+P$+",P,R"
380 PRINT"OPEN1,8,3,"B$(1)"0:"P$",P,R"B$(1)""
390 GET#1,A$,B$:IFASC(B$)<>4THENCLOSE1:STOP
```

```
400 IFB=0GOTO460
410 PRINTL$;:K=X:FORJ=BTO1STEP-1:PRINT" ";A$(J);:X$=A$(J)
420 X$=X$+L$
430 IFX$(K)>X$THENX$(K+J)=X$(K):K=K-1:GOTO430
440 X$(K+J)=X$:NEXTJ:X=X+B:PRINT:B=0
450 REM NUOVA LINEA E TEST DI FINE
460 GET#1,A$,B$:IFLEN(A$)+LEN(B$)=0GOTO760
470 REM GET NUMERO DI LINEA
480 GET#1,A$:L=LEN(A$):IFL=1THENL=ASC(A$)
490 GET#1,A$:A=LEN(A$):IFA=1THENA=ASC(A$)
500 C=C2:C1=-1:L=A*256+L:L$=STR$(L):IFLEN(L$)<6THENL$=LEFT$(S$,6-LEN(L$))+L$
510 REM GET BASIC STUFF
520 GET#1,A$:A=LEN(A$):IFA=1THENA=ASC(A$)
530 C9=C(A):IFC9>C1GOTO610
540 IFC2=6ANDLEN(M$)<5THENM$=" "+M$:GOTO540
550 K=0:IFB=0GOTO590
560 FORJ=1TOB:IFA$(J)=M$GOTO600
570 IFA$(J)<M$THENNEXTJ:K=B:GOTO590
580 FORK=BTOJSTEP-1:A$(K+1)=A$(K):NEXTK
590 B=B+1:A$(K+1)=M$
600 C=C2:C1=-1:M$=""
610 IFC2=5GOTO650
620 IFA=137ORA=138ORA=141ORA=167THENC=6:GOTO700
630 IFA=44ORA=32GOTO700
640 IFC9<>6THENC=9:GOTO700
650 IFC9=CTHENC=-1:C1=4
660 IFC>6GOTO700
670 IFC<0ANDC9>C1ANDC9>6THENC1=C9:GOTO690
680 IFC2=5THENIFLEN(M$)>2ORC>0GOTO700
690 M$=M$+A$
700 ONC9+1GOTO400,710,710,710:GOTO520
710 B$=B$(C9):C$=""
720 GET#1,A$:IFA$=""GOTO400
730 IFA$=B$GOTO520
740 IFA$<>Q$GOTO720
750 A$=B$:B$=C$:C$=A$:GOTO720
760 CLOSE1:INPUT"SU STAMPANTE     S■■■";Z$
770 C=3:Z=6:IFZ$="S"THENC=4:Z=12
780 OPEN2,C:PRINT#2:PRINT#2,"J"
790 PRINT#2,CHR$(1)"C R O S S   R E F E R E N C E"
800 PRINT#2:PRINT#2,"PROGRAMMA  :  ";CHR$(1)P$
810 PRINT#2:PRINT#2,MA$:PRINT#2,MB$:PRINT#2,MA$
820 X$="":MC$="          !":FORJ=1TOX:A$=X$(J)
830 IFC2=6THENK=6:GOTO850
840 FORK=1TOLEN(A$):IFMID$(A$,K,1)<>" "THENNEXTK:STOP
850 B$=LEFT$(A$,K-1):C$=MID$(A$,K+1):IFX$=B$GOTO870
860 PRINT#2,MC$:Y=0:X$=B$:PRINT#2,X$;LEFT$(S$,5-LEN(X$))" !";
870 Y=Y+1:IFY<ZGOTO890
880 Y=1:PRINT#2:PRINT#2,S$" !";
890 PRINT#2,LEFT$(S$,6-LEN(C$));C$;
900 MC$="":NEXTJ:PRINT#2:CLOSE2
```

C R O S S   R E F E R E N C E

```
PROGRAMMA : CROSS REFERENCE

------*-------------------------- - - -
VAR.  ! LINEA DEL PROGRAMMA
------*-------------------------- - - -
      !
A     !  490   500   520   530   620   630
A$    !  390   460   480   490   520   690   720   730   740   750   820
      !  840   850
A$(   !  250   410   560   570   580   590
B     !  400   410   440   550   560   570   580   590
B$    !  390   460   710   730   750   850   860
B$(   !  250   270   380   710
C     !  500   600   620   640   650   660   670   680   770   780
C$    !  710   750   850   890
C(    !  250   320   330   340   530
C1    !  500   530   600   650   670
C2    !  310   330   500   540   600   610   680   830
C9    !  530   640   650   670   700   710
J     !  320   330   410   430   440   560   570   580   820   900
K     !  410   430   440   550   570   580   590   830   840   850
L     !  480   500
L$    !  410   420   500
M$    !  540   560   570   590   600   680   690
MA$   !  230   810
MB$   !  240   810
MC$   !  820   860   900
P$    !  350   360   370   380   800
Q$    !  270   740
S$    !  270   500   860   880   890
X     !  410   440   820
X$    !  410   420   430   440   820   850   860
X$(   !  250   430   440   820
Y     !  860   870   880
Z     !  770   870
Z$    !  280   290   310   760   770
```

```
PROGRAMMA : CROSS REFERENCE

------*-------------------------- - - -
VAR.  ! LINEA DEL PROGRAMMA
------*-------------------------- - - -
      !
  280 !   300
  310 !   290
  350 !   360
  400 !   700   720
  430 !   430
  460 !   400
  520 !   700   730
  540 !   540
  590 !   550   570
  600 !   560
  610 !   530
  650 !   610
  690 !   670
  700 !   620   630   640   660   680
  710 !   700
  720 !   740   750
  760 !   460
  850 !   830
  870 !   850
  890 !   870
```

--*H*----*H*----*H*----*H*----*H*--

8010 Modem

Che cosa e' un MODEM?

Potrebbe sembrare una misteriosa scatoletta nera con due strane cavita'.

In realta', quest'aggeggio, e' appositamente designato per l'utilizzo di comunicazione fra due PET-CBM tramite la rete telefonica o interfonica.

La parola MODEM deriva da

MOdulator DEModulator

ed e' proprio con questo termine che e' conosciuto in tutto il mondo.

Il MODEM della Commodore modello 8010 e' una device di comunicazione a 300 boud. Le varie caratteristiche intrinseche dell'8010 fanno si da essere conforme allo standard internazionale CCITT.

Il sistema di utilizzo del MODEM 8010 e' semplicissimo:

Dopo aver composto il numero telefonico del corrispondente e aver scambiato alcuni messaggi verbali si appoggia la cornetta telefonica sul MODEM e dopo averla ben inserita nelle due apposite cavita' si puo' iniziare il colloquio fra i due PET.

Il MODEM 8010 e' connesso direttamente a qualsiasi serie di PET-CBM tramite il cavo a standard IEEE-488 e l'indirizzo di device corrisponde a 5. Con una semplice modifica si potra' cambiare l'indirizzo 5 in un altro eventualmente desiderato.

Il MODEM CBM 8010 e' composto essenzialmente in due parti di cui la prima e' costituita dall'alimentatore (input 220V AC. output 20V AC/400 mA).

La seconda parte e' il MODEM vero e proprio.

Sulla zona frontale del MODEM esistono due deviatori e quattro Leed (spie rosse luminose).

Un deviatore definisce la funzione del MODEM al momento dell'uso

ORiginate = il MODEM in questione trasmette l'onda portante

ANswer = il MODEM in questione e' abilitato alla funzione di risposta

La posizione intermedia di questo deviatore pone il MODEM in stato di spento.

Il secondo interruttore prevede, anch'esso, tre stati:

1 - FD = Full Duplex
Duplex completo
(comunicazione bilaterale)

2 - HD = Half Duplex
mezzo duplex
(comunicazione inilaterale)

3 - TST= TeST, posizione mediana
per prove in locale.

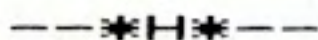
In qualsiasi posizione siano questi deviatori, con MODEM 8010 acceso naturalmente, la conferma del funzionamento e' convalidata dall'accensione di uno o piu' Leed posti fra gli interruttori.

Un perfetto accoppiamento fra due MODEM 8010 e' confermato dal Leed XMT o dal Leed RCV.

Il manuale, allegato alla confezione di questa apparacchiatura, e' costituito da 13 semplici pagine esplicative.
Per rendere facile il primo approccio con il MODEM 8010 vengono suggeriti due semplici programmi atti a iniziare dei collegamenti bilaterali.

Le caratteristiche tecniche del MODEM 8010 sono:

1 - Velocita' di trasferimento	300 boud 30 caratteri per secondo
2 - Compatibilita'	CCITT V.21
3 - Frequenze di trasmissione	OR Mark-Space 980-1180 Hz AN Mark-space 1650-1850 Hz
4 - Frequenze di ricezione	OR Mark-Space 1650-1850 Hz AN Mark-space 980-1180 Hz
5 - Stabilita' di frequenza	controllata a cristallo di quarzo +/- 0.3%
6 - Sensibilita' di ricezione	-50 dBm ON , -53 dBm OFF
7 - Livello di trasmissione	da -9 a -16 dBm
8 - modulazione	FSK (Frequency Shift Keyed)
9 - Tempo di ritardo di rivelazione	1.2 sec. ON , 120 msec. OFF
10 - Interfaccia	standard IEEE 488
11 - Condizioni di lavoro	da 10 a 40 gradi ambiente da 10% a 90% umidita' relativa
12 - Alimentazione	20V AC/0.4 A

--*H*--

Per concludere questa breve presentazione del MODEM 8010 della Commodore e' bene anche parlare di un ulteriore prodotto per il MODEM stesso.
Questo e' uno dei prodotti che fanno parte di:
-Approvato Harden Commodore-
e si chiama:

M C S
Modem Comunication System

L'MCS e' un firmware che rende il collegamento fra due o piu' PET, allacciati con il CBM 8010, molto piu' semplice e versatile.
Il firmware MCS e' residente su EPROM che deve essere inserita sullo zoccolo di integrato libero sulla piastra dei componenti del PET.

Per attivare l'MCS e' sufficente impartire il comando:

SYS (45056)

cio' abilitera' ben dieci comandi speciali per il MODEM.

Comando	effetto
RECEIVE	mette il PET in ricezione.
TXPAGE	trasmette le prime 23 righe d ello schermo.
TXPRG	trasmette il programma BASIC residente in memoria
TXLIN	trasmette un testo compreso f ra due -"-.
TXMEM	trasmette una zona definita d i memoria.
TXCOM	trasmette dei comandi diretti BASIC al PET in ascolto che li eseguira'.
TRCOM	come TXCOM e dopo la trasmissione si mette automaticamente in RECEIVE.
TXVAR	trasmette il contenuto di una variabile.
RCVAR	riceve il contenuto di una variabile.
HDCOPY	stampa le prime 23 righe dell o schermo.

L'MCS prevede un controllo alquanto sofisticato della trasmissione dei dati, a tal punto da ripetere per ben dieci volte, automaticamente, un messaggio se questo venisse ricevuto non correttamente a causa, ad esempio, di interferenze telefoniche.

Questo protocollo di collegamento firmwarizzato permette una grande flessibilita' dell'uso del PET in comunione con il MODEM CBM 8010.

La EPROM contenente l'MCS con alcuni fogli descrittivi e' reperibile presso i rivenditori Harden-Commodore specificando la versione del proprio PET (serie 3000 o serie 4000-8000).

UNSTRING S$ DELIMITED BY CC$

di
Gloriano Rossi

E' sempre mio uso prendere esempio e a paragone le capacita' del PET in confronto con i grossi sistemi, non solo per dimostrare che il nostro computer Commodore ha enormi capacita', ma per mettere a disposizione dei miei lettori, e quindi di possessori di CBM, alcune utility e particolari risoluzioni che possono dimostrarsi utilissime in molteplici casi.

Questa volta voglio parlarvi di una istruzione specifica di un linguaggio evoluto, il COBOL. L'istruzione e' l'UNSTRING.

A cosa serve questo comando in quel linguaggio di programmazione, e come utilizzarlo in BASIC sul PET?

Nel COBOL.

Nel COBOL ed in altri linguaggi evoluti le lunghezze dei records e dei campi relativi sono fissi. Occorre, infatti, predefinire campo per campo le varie dimensioni.

Tutto cio', se puo' essere comodo da una parte, rende difficoltoso il trattamento di determinati tipi di informazioni.

Facciamo un esempio classico.

Un record anagrafico, per determinate ragioni, deve essere lungo 80 caratteri; deve contenere un codice elettrocontabile, il codice fiscale, il nome, l'indirizzo, la localita' ed il CAP.

Analizziamo campo per campo le varie lunghezze.

Il CODICE ELETTROCONTABILE potrebbe avere una lunghezza fissa di tre caratteri.
Il CODICE FISCALE (tralasciamo il numero di partita IVA) e' lungo esattamente 16 caratteri.
Il NOME, mediamente, puo' essere lungo fino a 30 caratteri.
L'INDIRIZZO, come il nome,potra' avere 30 caratteri e cosi' pure per la LOCALITA'.
Infine il CAP, si sa', cinque caratteri.
Ora sommando le varie lunghezze (3+16+30+30+30+5) ci accorgiamo che avremmo bisogno di un record di ben 114 caratteri, senza contare poi che il NOME o l'INDIRIZZO o la LOCALITA' ognuno puo' superare i 30 caratteri definiti.

In COBOL si scriverebbe:

```
        01  ANAGRAFICA.
            02 CODICE           PIC 999.
            02 COD-FISCALE      PIC X(16).
            02 NOME             PIC X(30).
            02 INDIRIZZO        PIC X(30).
            02 CITTA            PIC X(30).
            02 CAP              PIC 99999.

                                 lunghezza
  nome variabile               (X=alfanumerica)
                               (9=numerica)
```

L'uso del comando UNSTRING permette al programmatore di rendere elastico un determinato numero di bytes per informazioni. Vediamo come si dovra' definire il record che da 114 caratteri e' passato ad 80 bytes.

```
        01  ANAGRAFICA.
            02 CODICE           PIC 999.
            02 COD-FISCALE      PIC X(16).
            02 MISTO            PIC X(56).
            02 CAP              PIC 99999.
```

Ed in un'altra zona di memoria si definiscono delle aree di lavoro transitorie:

```
        01  TRANSITORIE.
            02 NOME             PIC X(54).
            02 INDIRIZZO        PIC X(54).
            02 CITTA            PIC X(54).
```

Al momento della elaborazione si leggera' il record ad 80 caratteri e lo si porra' in quella "maschera" chiamata ANAGRAFICA e quindi si impartira' il comando:

```
    UNSTRING MISTO DELIMITED BY "*"
       INTO  NOME  INDIRIZZO  CITTA.
```

Se noi nel record appena letto avevamo esattamente questa serie di informazioni:

```
001RSSGLR46M30F605JGLORIANO ROSSI JUNIOR*CORSO PORTA NUOVA N.46*MILANO      20121
```

nelle variabili fino ad ora definite troveremo:

```
CODICE      = 001
COD-FISCALE = RSSGLR46M30F605J
MISTO       = GLORIANO ROSSI JUNIOR*CORSO PORTA NUOVA N.46*MILANO

NOME        = GLORIANO ROSSI JUNIOR
INDIRIZZO   = CORSO PORTA NUOVA N.46
CITTA       = MILANO
```

Avremo ottenuto una minore occupazione nella memorizzazione su disco ed una maggiore elasticita' delle informazioni.

In BASIC.

In BASIC, si sa', una variabile e' lunga quanto e' lungo il suo argomento, ma per particolari tipi di gestione e' comodo manipolare le informazioni di lunghezza prefissata.

Un classico esempio di questa necessita' e' proprio l'obbligatorieta' di lunghezza fissa dei records scritti in files ad organizzazione tipo RANDOM o RELATIVE.

Un altro esempio e' quello di voler scrivere un record costituito da una sola variabile a 80 caratteri contenente poi tutte le informazioni desiderate.

In BASIC tratteremo questo record nella maniera seguente:

```
nn00 REM RR$ = RECORD DA 80 CARATTERI
nn10 CO$ = LEFT$(RR$,3)    :REM CODICE
nn20 CF$ = MID$(RR$,4,16) :REM COD-FISCALE
nn30 S$ = MID$(RR$,21,56) :REM MISTO
nn40 CP$ = RIGHT$(RR$,5)  :REM CAP
```

Fermo restando il concetto che se si conoscono le caratteristiche fisiche di una informazione o di una serie di informazioni, la gestione delle stesse risulta inevitabilmente piu' facile ed elastica, questa routine sara' utile per risolvere alcuni problemi di gestione del record.
L'UNSTRING che propongo analizza una variabile alfanumerica e la scinde in tante altre quante volte e' riportato il carattere di controllo (generalmente l'asterisco).

Cio' che vi propongo pero' e' un programma completo che e' diviso essenzialmente in due parti.
La prima parte contiene una serie di istruzioni atte alla dimostrazione della validita' della routine stessa.
La seconda parte, quella numerata da 60000 in poi, e' la routine UNSTRING vera e propria che dovra' essere inserita in un qualsiasi vostro programma che necessiti di questa funzione.

```
100 NC=10:DIM AR$(NC)           :REM  NC = NUMERO MASSIMO CAMPI DIVISI DA CC$
110 INPUT "LA STRINGA ";S$      :REM INTRODUZIONE DELLA STRINGA CON ASTERISCHI
120 GOSUB 60000                 :REM RICHIAMO ROUTINE UNSTRING
130 FOR I=1 TO NC               :REM  !
140 PRINT AR$(I)                :REM  + STAMPA NC CAMPI
150 NEXT I                      :REM  !
160 END                         :REM FINE PROGRAMMA "

60000 REM ***************************************************************
60050 REM *****                                                    *****
60100 REM *****    UNSTRING  S$ DELIMITED BY CC$ INTO  AR$(NC)      *****
60120 REM *****                                                    *****
60140 REM ***************************************************************
60150 AR=1                      :REM INDICE DI CAMPO
60160 CC$="*"                   :REM CARATTERE DI CONTROLLO
60170 K=0                       :REM INDICE FLOTTANTE
60180 FOR J=K+1 TO LEN(S$)
60190 IF MID$(S$,J,1) <> CC$ THEN GOTO 60210
60200 NEXT J
60210 IF J>LEN(S$) THEN GOTO 60270
60220 FOR K=J TO LEN(S$)
60230 IF MID$(S$,K,1) = CC$ THEN GOTO 60250
60240 NEXT K
60250 AR$(AR) = MID$(S$,J,K-J) : AR=AR+1
60260 IF K<LEN(S$) THEN GOTO 60180
60270 RETURN
```

E' inutile che io riporti una routine inversa alla UNSTRING, la STRING, in quanto sara' sufficente costruire una stringa inserendo nel vostro programma un qualche cosa di simile a quasta serie di istruzioni che per prova potranno essere inserite nel prog. di dimostrazione.

```
10 FOR I=1 TO 56: B$ = B$ + " ": NEXT I  :REM CREA UNA STRINGA A SPAZIO "

110 CC$="*": S$=""
111 FOR I=1 TO NC
112 PRINT "CAMPO " I; :INPUT X$
113 S$ = S$ + X$ + CC$
114 NEXT I
115 S$ = LEFT$(S$,LEN(S$)-1): REM TOGLIE L'ULTIMO ASTERISCO INDESIDERATO
116 IF LEN(S$) < 56 THEN S$=S$ + LEFT$(B$,56-LEN(S$))
117 IF LEN(S$) > 56 THEN PRINT "TROPPO LUNGA": GOTO 110
```

```
CAMPO  1  GLORIANO      CAMPO  4  PORTA       CAMPO  7  46
CAMPO  2  ROSSI         CAMPO  5  NUOVA       CAMPO  8  MILANO
CAMPO  3  CORSO         CAMPO  6  N.          CAMPO  9  20121

GLORIANO*ROSSI*CORSO*PORTA*NUOVA*N.*46*MILANO*20121*-

CAMPO  1  = GLORIANO    CAMPO  4  = PORTA     CAMPO  7  = 46
CAMPO  2  = ROSSI       CAMPO  5  = NUOVA     CAMPO  8  = MILANO
CAMPO  3  = CORSO       CAMPO  6  = N.        CAMPO  9  = 20121
                                              CAMPO  10  = -
```

Come trasformare, in casa, il nostro PET in 4032

Gia' nel numero scorso di POCKET PET avevo annunciato la disponibilita' delle ROM del BASIC 4.0 da sostituire a quelle del BASIC 3.0.

Tutti i PET 2001, nuove ROMs, fino ai PET-CBM 3032, possono essere aggiornati con la nuova versione del BASIC, di gran lunga piu' potente e veloce.

Dopo aver acquistato, dal proprio rivenditore autorizzato Harden-Commodore di fiducia, il kit di cinque ROMs, si deve sostituire le quattro precedenti con le nuove.

Molti di Voi non hanno aperto ancora il proprio computer PET-CBM 3032.

Bene e' ora di farlo!

Prima di fare qualsiasi cosa occorre ricordarsi di staccare la spina della corrente.

Disponete il PET-CBM su un tavolo con spazio abbastanza ampio di lavoro.

Procuratevi un cacciavite a stella.

Puo' andare bene anche quello della borsa degli attrezzi della automobile.

Afferrate ora il PET da parte del lato anteriore (il bordo della tastiera) e ribaltatelo all'indietro fino a che non appoggi saldamente sul dorso.

La parte sottostante al PET, quella nera, dovrebbe essere ora rivolta verso di voi.

Questa parte metallica e' solidale con la parte superiore per mezzo di due viti a stella che sono inserite attraverso due linguette negli angoli destro e sinistro della parte inferiore. A PET ribaltato, le viti, risulteranno in alto a destra e a sinistra.

Quando avrete individuato le viti rimuovetele e mettetele da parte per il momento.

Le parti superiore ed inferiore del PET non sono ora piu' fissate fra di loro in corrispondenza della parte frontale.

Tenete insieme queste parti con le mani e riportate l'intero computer nella sua posizione normale.

Sollevate ora la parte superiore del PET che si aprira' a ventaglio per mezzo di un cardine posto in fondo.

Dando una occhiata all'interno, in qualche posto c'e' una barretta metallica lungo il fianco od il bordo del vostro PET.

Questa barra servira' come sostegno della parte superiore proprio come il cofano di qualsiasi autovettura.

All'interno del PET troverete molte cose.

Quella che vi interessa, pero', e' la piastra dei componenti, cioe' quella scheda piatta a circuito stampato fissata sul fondo nero su cui sono alloggiati molti componenti elettronici, alcuni dei quali sono inseriti su zoccolini generalmente bianchi.

Alcuni di questi componenti si chiamano circuiti integrati e, nell'aspetto, sono uguali per forma,ed alcuni anche per dimensione, a quelli che avete appena comperato.

Quali sono i quattro integrati da togliere e quali sono le giuste posizioni di quelli nuovi?

Non e' difficile individuarli.

Esite, sulla piastra dei componenti, una fila orizzontale di zoccolini, tre dei quali sono vuoti.

Noterete che sul circuito stampato e' disegnato in corrispondenza di questi zoccoli vuoti i termini:

UD3, UD4 e UD5

Quando avrete individuato la fila interessata, togliere i curcuiti integrati contrassegnati con:

UD6, UD7, UD8 e UD9

notando che ogni integrato possiede una tacchettina rivolta verso di voi.

Avremo a questo punto sette zoccolini vuoti.

Prendere ora, con massima cura, il kit dei cinque nuovi integrati e con massima delicatezza ed attenzione, soprattutto per i piedini delle ROMs, si inseriscono questi componenti in queste esatte posizioni, tenendo presente che le tacche di ogni integrato deve essere sempre rivolta verso di voi.

Per fare queste operazioni puo' essere necessario allineare i piedini con i relativi fori degli zoccoli. Per fare cio' utilizzate delle pinzette ed eseguite questa azione solo se necessario e con cura.

circuito integrato	posizione
libero	UD3
libero	UD4
901465-23	UD5
901465-20	UD6
901465-21	UD7
901447-29	UD8
901465-22	UD9

Per ogni circuito integrato assicuratevi che la tacca sia nella giusta posizione e che tutti i piedini abbiano imboccato il rispettivo foro. E' facile infatti mancarne uno e piegare il relativo piedino all'esterno o verso l'interno.

Iniziate a premere delicatamente fino a che ogni integrato non sia completamente inserito.

Quando avrete eseguito tutte queste operazioni eseguite ancora una volta una verifica di tutti i piedini e delle relative posizioni degli integrati e quindi potete procedere a chiudere il vostro "nuovo" PET, eseguendo in ordine inverso le operazioni che avete compiuto per aprirlo.

Inserite di nuovo la spina nella presa di corrente, ed appena acceso apparira':

```
*** COMMODORE BASIC 4.0 ***
    nnnnn BYTES FREE
    READY
    ░
```

e come trasformare,

sempre in casa,

il nostro

2040 o 3040 in 4040

cioe' il DOS 1.0 in DOS 2.0 ??

DOS 1.0 in DOS 2.0

Per poter trasformare la propria unita disco CBM 2040 o CBM 3040 in nuova device corrispondente a quella oggi in commenrcio, la CBM 4040, e' sufficente reperire presso il proprio rivenditore autorizzato Harden-Commodore di fiducia, il kit di ROMs del DOS 2.0.

Quando avrete fatto cio', noterete che il kit e' composto di tre circuiti integrati uguali a quelli del BASIC 4.0, e di un integrato molto piu' grande.

Come si procede alla trasformazione?

Se avete gia' trasformato il vostro PET , le operazioni che seguono sono alquanto semplici.

Il coperchio dell'unita' 2040 o 3040 e' sollevabile quando si saranno tolte le due viti di fissaggio poste sulle due fiancate dell'apparacchiatura. Non sara' quindi necessario alcun ribaltamento come e' accaduto per il PET.

Dopo aver aperto il coperchio si potranno notare tutte le parti che compongono l'unita' dischi. La piastra dei componenti che ci interessa e' fissata proprio sul coperchio stesso.

Su questa si potra' notare che un solo zoccolo e' libero ed a fianco a questo ci sono due circuiti integrati simili a quelli da sostituire.

Con sicurezza potrete togliere sia l'integrato che sta' sulla destra e quello che si trova a sinistra dello zoccolo vuoto, cosi' da ottenere tre zoccolini liberi. Ricordarsi la posizione della tacca, che dovra' essere rispettata anche con i nuovi integrati.

Ora potremo ricercare l'"integratone" da togliere. Questo si trova piu' o meno al centro della piastra dei componenti e dovrebbe avere una siglatura corrispondente a -901466-02-.

Per fugare qualsiasi dubbio questo integrato si trova in posizione UK3 che e' fra l'integrato 6522 (posto in posizione UM3) ed il 6504 (posto in posizione UH3), quest'ultimo leggermente piu' piccolo degli altri due.

Dopo aver tolto i tre circuiti integrati si procede a riempire tutti i zoccoli vuoti con i componenti che del kit che avete appena acquistato.

circuito integrato	posizione
901468-12	UL1
901468-11	UJ1
901468-13	UH1
901466-04 o 901466-34	UK3

L'integrato "grosso" potrebbe avere una siglatura terminante in 04 o 34, la differenza e' da considerarsi insignificante in quanto non esiste alcuna variazione se non solamente nella sigla.

Inserire questi nuovi componenti rispettando le medesime regole che abbiamo seguito per la trasformazione del BASIC 3.0 in BASIC 4.0, e tenendo anche presente che tutte le tacche degli integrati devono sempre essere rivolte verso il basso.

Notizie

Ci e' giunta notizia da:

I S N E V
ISTITUTO PER LO STUDIO DELLA NEVE
E DELLE VALANGHE

Via Polonghera, 8
10138 TORINO
Tel. 011-442310/4471209

Da poco meno di due anni e' in funzione presso l'ISNEV un calcolatore elettronico modello CBM nella configurazione tipo di unita' centrale, video e tastiera (32K), floppy disk e stampante CBM.

Tale elaboratore oltre alle attivita' gestionali ed amministartive dell'Istituto, viene usato nel campo scientifico, nel campo dell'analisi delle caratteristiche del manto nevoso e della previsione delle valanghe.

In particolare con appositi programmi si effettua:

1 - Archiviazione dei dati meteonivometrici.

2 - Ordinamento dei dati meteonivometrici.

3 - Rappresentazione grafica mediante istogrammi dei dati archiviati.

4 - Previsioni della caduta di valanghe..

1 - Archiviazione dei dati meteonivometrici.
Tale attivita' prevede l'archiviazione dei dati contenuti in un modello standard previa verifica di congruenza sui singoli valori secondo una codificazione.
I dati corretti vengono immagazzinati sui dischetti e vanno a comporre l'erchivio base.

2 - Ordinamento dei dati meteonivometrici.
Prevede la riordinazione degli archivi sia per anni di rilevamento che per singola stazione.

3 - Rappresentazione grafica mediante istogrammi dei dati archiviati.

4 - Previsioni della caduta di valanghe.
Utilizzando l'archivio base e gli archivi secondari si procede poi a tracciare, utilizzando la stessa stampante CBM, una rappresentazione grafica delle grandezze mediante istogrammi che permettono, unitamente a un'altra serie di programmi piu' complessi, la sperimentazione di nuove metodologie di previsione della caduta di valanghe.

Molto utili per queste ultime elaborazioni sono i caratteri speciali che si hanno a disposizione usando la stampante originale CBM.

Dr. ing. Piermichele Balzaretti.

Balzaretti Piermichele

PET POSTA

Rubbrica aperta a tutti i lettori.

Poco tempo fa mi e' giunta in redazione un letterina di Mauro Manzoni di Montefiascone in provincia di Viterbo, e che pubblico con molto piacere.

Carissimo POKET PET,

ti mando una piccolissima cosa fatta in 'casa Non ha pretese, ma vuol essere un motivo di collegamento con te e con gli amici del PET. E' possibile pubblicare un programma per mettere in ordine alfabetico una lista di nomi? Non riesco a concretizzarlo in modo soddisfacente. Ti ringrazio e ciao.

Mauro - Montefiascone VT

```
100 PRINT"":A$="QQQQQQQQQQQ"
110 PRINTA$"";:PRINT" ";
120 B$="                                        "
130 B$=B$+"QUESTO PROGRAMMINO E' FATTO PER IL NOSTRO PET CON L'INTENTO DI"
140 B$=B$+" FAR EVIDENZIARE UN QUALSIASI TESTO SCORREVOLE SU UNA FASCIA"
150 B$=B$+" DEL MONITOR. ATTUANDO MODIFICHE SI PUO' ABBELLIRE A PIACERE...CIAO
"
160 FORJ=1TO241:FORI=1TO100:NEXT
170 PRINTA$""LEFT$(B$,40)
180 B$=RIGHT$(B$,235)+" ":NEXT
190 END
```

Una prima, semplice modifica si può attuare cambiando le linee 110 e 170 in:

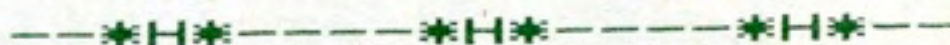
```
110 PRINTA$"";:FORI=1TO200:PRINT" ";:NEXT

170 PRINTA$""LEFT$(B$,40)
```

--*H*--

Caro Mauro,

ti ringrazio per la tua piccola collaborazione che spero non si fermi qui.

In aggiunta alla tua lettera direi di modificare le linee 160 180 e 190 del tuo programma in maniera tale da non essere legati dalla lunghezza di B$ ed in oltre per fare in modo che la tua dicitura continui in modo ciclico lo scorrimento.

```
160 FOR I = 1 TO 100 : NEXT    :REM TEMPO DI PAUSA
170 PRINT A$"" LEFT$(B$,40)
180 B$ = RIGHT$(B$,LEN(B$)-1) + LEFT$(B$,1) :REM ROTAZIONE DI STRINGA
190 GOTO 160
```

Almost a year ago.
The World's First Commodore PET Show
was held at the Cafe Royal.
The Second International Commodore PET Show
will be even bigger and better.

Whatever your profession may be,
if you are thinking of computerising your business,
the Commodore PET Show will prove to you,
that Europe's best selling microcomputer,
the Commodore PET Series,
has a solution to virtually every application.
Whether Industrial, Commercial, Medical, Educational,
or simply a computer enthusiast,
over 100 exhibitors from all over the world will be able to
demonstrate to you an application to suit your exact requirement.
In fact, everything a computer is capable of doing, the PET does it!
So come and see for yourself,
after all
over 100.000 European PET users can't be wrong.

Commodore Business Machines
will demonstrate their entire product line
and a series of seminars for specific user interest groups
has been timed to coincide with the show.

The Second International Commodore PET Show
West Centre Hotel, London SW6

Thursday 18th June 1.00pm - 7.00pm
Friday 19th June 10.00am - 7.00pm
Saturday 20th June 10.00am - 5.00pm

The Second International Commodore PET Show

A Londra, da giovedi' 18 e sabato 20 giugno di quest'anno, si e' svolto:

Il secondo PET Show internazionale della Commodore.

E', senza ombra di smentita, l'unico mini/personal computer a cui e' stata dedicata esclusivamente una manifestazione europea di tale portata.

Sicuramente erano presenti alla manifestazione piu' di un centinaio di espositori che mostravano piu' di un migliaio di prodotti software e hardware per il PET.

La casa madre, la Commodore Businnes Machines, ha colto l'occasione di presentare ufficialmente il CBM 8096, un nuovo computer della serie 8000, con 96K di memoria.

Il nuovo computer Commodore ha un video ad 80 colonne come l'8032, e prevede una grande capacita di caricamento di programmi.
Sara' possibile adattare una ampia vastita' di programmi speciali utilizzati nella gran parte di sistemi computer (programmi scritti nei linguaggi tipo FORTRAN o COBOL, oppure sofisticati DATA-BASE).

Verranno eseguite delle versioni aggiornate del VisiCalc, dell'OZZ e del Wordcraft.

Una altra novita di casa PET sara' una nuova unita' disco da 3.2 Megabytes (CBM 8062); questa device portera' il nuovo sistema ad alti livelli di sofisticazione.

La Harden spa ha gia' opportunamente preso i dovuti accordi al fine di poter commercializzare il sistema 8096 anche in Italia.

Sabato 20 a chiusura della manifestazione si e' tenuta una conferenza stampa atta a rendere edotti gli interessati sui nuovi studi sui prodotti del prossimo futuro. Il piu' importante progetto descritto e' risultato, senza dubbio, il MMF 9000 (MicroMainFrame); un "qualche cosa" che ha lasciato letteralmente a bocca aperta tutti gli astanti: un super mini computer. Si puo' dire che la Commodore con il MMF 9000 vuole soddisfare una larga fascia di esigenze di computerizzazione.

Sui prossimi numeri di POCKET PET descrivero' alcuni dei prodotti di particolare interesse presenti alla manifestazione, fra i quali posso annunciare il MUPET ed il COBOL per il PET. Il primo permette il collegamento di piu' PET ad una medesima unita' disco. Il secondo e' un sistema che trasforma il PET in un computer programmabile in linguaggio COBOL.

Proprio per questi prodotti in particolare e per molti altri, che descrivero' con maggior dettaglio rispetto questa breve nota di viaggio, la Harden spa ha intrapreso gli opportuni accordi commerciali.

In Libreria

HARDEN S.p.A. commodore

Sul numero zero di POCKET PET avevo presentato alcuni volumi di interesse generale.
Ora su questo numero sono lieto di presentare alcuni volumi dedicati esclusivamente ai prodotti della Commodore.

32 programmi con il PET

Franco Muzzio & C. editore ci presenta un libro di ben 240 pagine intitolato:

32 programmi con il
PET

il cui contenuto sono appunto trentadue programmi documentati rispettivamente con molte specifiche.
Ognuno dei 32 capitoli non si propone esclusivamente di fornire una semplice spiegazione del programma, al contrario questo e' corredato di:

1 - Scopo : ampia descrizione e significato del programma.
2 - Come usarlo : vengono spiegati dettagliatamente i modi di esecuzione.
3 - Listato : Listato completo del programma pronto per essere eseguito su qualsiasi tipo di PET (dalla serie 2000 vecchie ROM alla serie 8000).
4 - Esecuzione di prova : in ogni capitolo e' compresa una serie di fotografie riproducenti lo schermo del PET durante l'esecuzione del programma.

A questo punto, il capitolo, potrebbe terminare cosi', ma e' proprio cio' che segue che fa di questo libro un valido aiuto per il principiante ed una ottima "palestra" per chi non e' piu' un iniziato.

5 - Semplici variazioni : vengono suggerite alcune modifiche al fine di poter cambiare o migliorare il programma.
6 - Routines principali : ogni programma e' diviso in una serie di istruzioni. Ogni serie e' chiamata routine e proprio in questo sottocapitolo ne viene fornita la spiegazione.
7 - Variabili principali: come per le routines anche le varie variabili principali vengono elencate e descritte.
8 - Progetti suggeriti : quale conclusione di ogno capitolo vengono suggerite alcune idee per spingere il lettore a modificare o migliorare con le proprie forze il programma in oggetto.

E' inutile dire che tutti i programmi riportati girano regolarmente.

Il libro viene venduto presso tutte le librerie o direttamente presso l'editore.
Per favorire tutti i possessori di PET-CBM, la Harden spa ha acquistato

un gran numero di copie del volume; e' quindi possibile reperire questa edizione anche presso tutti i rivenditori autorizzati Harden-Commodore.

Il costo del volume e' di L.9.500

La Franco Muzzio editrice ci comunica che sono disponibili i 32 programmi su supporto magnetico direttamente presso la casa editrice. Su supporto a cassetta il costo corrisponde al L.20.000, mentre su dischette l'onere e' di L.25.000.
L'indirizzo dell'editore e':

Franco Muzzio editrice
via Bonporti 36
35100 Padova.

--*H*--

impariamo a programmare in BASIC con il PET / CBM

della professoressa Rita Bonelli, edito dal gruppo editoriale Jackson.

Questo libro, in elegante formato tascabile, ha le medesime dimensioni del POCKET PET che state leggendo, contiene ben 180 pagine.
Il contenuto del volume e' una quantomeno completa trattazione di cio' che e' e cio che si puo' fare con il PET-CBM serie 2000, vecchie ROM, e serie 2000, nuove ROM-BASIC 3.0.
La professoressa Rita Bonelli non si limita a descrivere l'unita' centrale Commodore, ma tratta anche tutti gli argomenti inerenti alla stampante CBM 3022, nonche' alla unita' disco CBM 3040 con versione DOS 1.0.
Un vero primo "Corano" del PET, dunque, consigliabile anche ai possessori di PET-CBM serie 4000 e 8000, perche' "la Bonelli" non si limita a descrizioni e all'uso del PET, ma si prolunga i concetti di programmazione fondamentale BASIC specifici del PET e suggerisce routines e trucchetti utili anche ai programmatori smaliziati.
Il volume e' disponibile presso tutti i rivenditori autorizzati Harden-Commodore, nonche' presso la maggior parte delle librerie. Se nonostante cio', vi fossero delle difficolta' di approvvigionamento e' possibile richiederlo direttamente alla Harden spa oppure alla casa editrice.
Il costo? Semplicemente L.10.000.

--*H*--

Proprio in corrispondenza dell'uscita del VIC 20 appare anche un volume dedicato interamente a questo nuovo personal della Commodore.
Una stretta collaborazione fra il personale della Harden spa, che ha fornito un prototipo ante-serie con documentazione varia, e la professoressa Rita Bonelli, ha fatto si' che fosse possibile realizzare questa edizione in concomitanza con l'entrata sul mercato del VIC 20.
Il volume e' stato dedicato al geom. Luigi Bonezzi, presidente della Harden spa, che purtoppo e' venuto a mancare nel mese di agosto di quest'anno.
Il volume, della medesima collana del precedente della Bonelli, si intitola:

Vogliamo
Incominciare
Cosi'
Impariamo a programmare
in BASIC
con il VIC / CBM

L'edizione contiene ben 176 pagine ricche di nozioni e di indispensabili consigli per il perfetto uso del VIC 20.
Una curiosita' su questo libro: tutto il testo e' stato impostato e ribrodotto tramite una stampante della Commodore pilotata dal programma WP 3.1,

proprio come il POCKET PET che state leggendo.
Per acquistare questa edizione valgono naturalmente le "regole" del precedente volume.

Il prezzo? al momento di andare in stampa con POCKET PET non mi era pervenuto ancora l'esatto ammontare, ma prevedo che manterra' il medesimo prezzo del primo della Bonelli. (L.10.000)

---*H*--- ---*H*--- ---*H*---

Lo Sparacaratteri

di Riccardo Saetti

Non e' facile spiegare al lettore la ragione di esistere di un simile programma.

Innanzitutto non ha alcuna pretesa di essere utile.
Non costituisce una novita' di software.

E' semplicemente un piccolo innocuo divertimento, un gadget all'ennesima potenza.
A cosa serve vedere e sentire, una stringa di caratteri mentre viene letteralmente sparata sullo schermo del PET?

Il punto focale del programma e' costituito dalla routine che crea l'effetto sonoro, molto simile a quello realizzato dai musicisti "pop".

Dopo che il lettore avra' dettato la stringa di caratteri desiderata, il programma provvede a stamparla, un carattere per volta pero', eseguendo (eccetto lo spazio), quale accompagnamento, il tipico suono dello sparo.

Non vi resta che provare l'effetto che fa a vedere "sparate" sul video le lettere che compongono il vostro nome o qualunque altra fase vogliate.

REMarks.

- 100 Azzera le locazioni del generatore dei suoni.
- 110-130 Input della stringa da "sparare". Se detta stringa risultasse piu' lunga di 36 caratteri avviene un rifiuto.
- 140 Prepara le locazioni del generatore di suoni.
- 150 Prepara la stringa cornice
- 160 Stampa la lettera "H" di Harden, stilizzata, e disegna la cornice.
- 170 Stampa un carattere. Se il carattere corrisponde ad uno spazio salta la ruotine di suono.
- 180 Routine del suono. L'effetto generato corrisponde ad un qualche cosa che velocemente scivola nell'aria.
- 190-210 Attende che sia battuto uno spazio per continuare.
- 220 Stampa la "H" di Harden in maniera stilizzata.

Modifiche.

Per poter utilizzare il programma anche su PET 4000 o 800 modificare:

200 IF PEEK (151) = 32 THEN 100

```
10 REM*************************************************
20 REM***   POCKET  GROUP    ***   RICCARDO SAETTI    ***
30 REM*************************************************
40 REM"
100 FORI=59464TO59467:POKEI,0:NEXT
110 GOSUB220:PRINT"▒▒▒▒▒▒▒▒▒INSERISCI UNA STRINGA LUNGA AL MASSIMO"
120 PRINT"36 CARATTERI."
130 INPUTA$:IFLEN(A$)>36THEN110
140 POKE59468,12:POKE59464,0:POKE59467,16:POKE59466,15
150 W$="*":FORX=1TO40:W$=W$+"*":NEXT
160 GOSUB220:PRINT"▒▒▒▒▒▒▒▒▒▒"W$;SPC(38)"**"SPC(38)"**"SPC(38);W$"TT▶▌";
170 FORX=1TOLEN(A$):Q$=MID$(A$,X,1):PRINTQ$;:IFQ$=" "THEN190
180 FORY=50TO200STEP3:POKE59464,Y:NEXT:POKE59464,0:FORY=1TO100:NEXT
190 NEXT:PRINT:PRINTSPC(254)"▒▒▒PREMI ▒SPACE▒";
200 IFPEEK(151)=6THEN100
210 GOTO200
220 PRINT"▒▬▬ ▬▬":PRINT" ▒ ▬▬▒ ▬":PRINT"▬▒ ▬ ▬▒ ▬":RETURN
```

ST DS DS$

Quando noi trattiamo dei files, di qualsiasi natura essi siano, possiamo incorrere in errori o in condizioni particolari che in ogni caso occorre conoscere per una giusta gestione dei dati.

Utilizzando una od entrambe le tape cassette (modello C2N) si possono verificare otto stati differenti. Questi stati vengono comunemente chiamati rapporti di File Status.

Una variabile, che non deve mai essere usata per altri scopi, contiene sempre un valore che ci puo' dire in che condizione di stato ci troviamo.

Questa variabile e' sempre numerica e si chiama ST.

Ogni volta noi trattiamo un file e' consigliabile eseguire un test di questa variabile e se il suo contenuto soddisfa determinate condizioni si procede in relazione del significato relativo.

Cosi' ad esempio ogni volta che eseguiamo una OPEN di file su cassetta e' bene seguire con il test:

```
IF ST <> 0 THEN ...
```

Nella stessa maniera si procede anche per la lettura o scrittura di ogni

valore numerico di S T	lettura su registratore unita' 1 o unita' 2	verifica e load da registratore unita' 1 o unita' 2	lettura/scrittura o verifica su devices IEEE 488 (dischi ecc.)
1			fuori tempo in fase di scrittura
2			fuori tempo in fase di lettura
4	Blocco corto	Blocco corto	
8	Blocco lungo	Blocco lungo	
1 6	Errore irrecuperabile	Mancato accoppiamento	
3 2	Errore di parita'	Errore di parita'	
6 4	Fine del file		Fine identificatore (fine file)
- 1 2 8	Fine nastro	Fine nastro	Device non presente

record, questo non solo per sapere se tutto e' andato OK, ma anche, in fase di lettura in particolare, se ci troviamo alla fine del file (ST=64).

Le condizioni di errore vere e proprie per la gestione delle cassette sono in realta' solamente quattro (ST= 4 , 8 , 16 , 32), mentre le condizioni 0 , 64 , 128 sono considerate messaggi di normale gestione.

La variabile del File Status puo' essere adoperata anche nella gestione dei dischi. Infatti avremo un messaggio comune alla gestione cassette, ST=64 = fine del file, mentre lo status 1 e 2 ci potra' dire che il cavo PET/IEEE e' troppo lungo. Lo ST=128 corrispondera' ad uno stato di mancanza di device; provate ,per dimostrare cio eseguire ad esempio:

```
OPEN 1,14: PRINT ST
```

A seguito di questa piccola disertazione riportiamo la tabellina dei valori della variabile ST e dei relativi significati che essa assume in conseguenza della device che nel momento del test e' interessata.

Per la gestione dei dischi, pero', esiste' una'altra serie di messaggi altrettanto utili.
Per le versioni BASIC inferiori alla 4.0, questa serie di messaggi e' consultabile solamente in questa maniera:

```
OPEN 15,8,15
INPUT#15 ,A,B$,C,D
PRINT A,B$,C,D
CLOSE 15
e quindi:
IF A <> 0 THEN ...
```

Con le versioni BASIC 4.0 e seguenti tutto cio' e' estremamente semplificato; infatti e' sufficente impostare:

```
PRINT DS o IF DS .......
        oppure
PRINT DS$
```

Il secondo caso per ottenere il messaggio completo.

Con le versioni BASIC 4.x, allora, abbiamo altre due variabili vietate all'uso comune, ma in ogni caso esiste un grande risparmio di tempo e di scrittura.
E' bene, per questa ragione, che tutti i programmi che vengono trasportati dalle vecchie versioni BASIC al 4.x vengano analizzati al fine che non contengano tali variabili per altri scopi.

Per facilitare questo esame e' consigliabile l'uso del programma CROSS REFERENCE descritto e proposto in questo numero di POCKET PET.

Ecco qui a seguito la tabellina dei valori e significati delle variabili DS e DS$.
Come noterete questa tabella e' completa anche dei messaggi relativi agli errori inerenti alle versioni BASIC 4.x e DOS 2.x.

Disk Status

Con BASIC 3.0 e 4.x

```
OPEN 15,8,15
INPUT#15 ,A,B$,C,D
PRINT A,B$,C,D
CLOSE 15
```

Con il solo BASIC 4.x

```
PRINT DS
   o
PRINT DS$
```

ed otteniamo in entrambi i casi:

```
 n n    messaggio errore    t t    s s
-----  ------------------  -----  -----
  !            !             !      !
  !            !             !      *----- riferimento settore
  !            !             *------------ riferimento traccia
  !            *-------------------------- tipo errore
  *--------------------------------------- codice errore
```

	numero errore	messaggio errore	traccia	settore
messaggi di stato	00	OK	00	00
	01	File/s scratciato (cancellato)	#FILE	00
READ ERRORs - errori di lettura	20	READ ERROR (intestazione di blocco mancante)	T	S
	21	READ ERROR (non esiste carattere di sincronismo)	T	S
	22	READ ERROR (non esiste blocco dati)	T	S
	23	READ ERROR (errore di parita')	T	S
	24	READ ERROR (errore di decodifica)	T	S
	27	READ ERROR (errore di parita' nella intestazione di blocco)	T	S
WRITE ERRORs - errori di scrittura	25	WRITE ERROR (scrittura o verifica)	T	S
	26	WRITE PROTECT ON (il disco e' protetto contro la scrittura)	T	S
	28	WRITE ERROR (blocco troppo lungo)	T	S
	29	DISK ID MISMATCH (l'identificatore del disco non corrisponde o inesistente)	T	S
SYNTAX ERRORs - errori di sintassi	30	SYNTAX ERROR (errore generalizzato)	00	00
	31	SYNTAX ERROR (comando invalido)	00	00
	32	SYNTAX ERROR (linea troppo lunga)	00	00
	33	SYNTAX ERROR (nome file invalido)	00	00
	34	SYNTAX ERROR (impostazione del nome del file non corretta)	00	00
	39 *	SYNTAX ERROR (comando DOS invalido)	00	00
	50 *	SYNTAX ERROR (non esiste Record)	00	00
	51 *	SYNTAX ERROR (Record troppo grosso)	T	S
	52 *	SYNTAX ERROR (File troppo grande)	T	S
FILE ERRORs - errori sui Files	60	WRITE FILE OPEN (si tenta una scrittura su un file gia' aperto)	00	00
	61	FILE NOT OPEN (si tenta di scrivere su un file non aperto)	00	00
	62	FILE NOT FOUND (non esiste file)	00	00
	63	FILE EXISTS (si tenta di creare un file gia' esistente)	00	00
	64	FILE TYPE MISMATCH (e' stato omesso il tipo di 'file)	00	00
	65	NO BLOCK (il blocco richiesto non e' disponibile)	T	S
	66 *	ILLEGAL TRACK AND SECTOR (si tenta di accedere ad una traccia e settore inesistente)	T	S
	67 *	ILLEGAL SYSTEM TRACK AND SECTOR (si tenta di utilizzare una traccia e settore riservati al sistema)	T	S
SYSTEM ERRORs - errori di sistema	70	NO CHANNEL (non esiste il canale)	00	00
	71	DIR ERROR (errore sulla directory)	00	00
	72	DISK FULL (disco pieno)	00	00
	73 *	DOS MISMATCH (incompatibilita' DOS)	00	00
	74 *	DRIVE NOT READY (si tenta di accedere ad una unita' 8050 non presente)	00	00

gli errori segnati con -*- sono presenti con le versioni DOS 2.x

Harden non vende solo computers. Vende soluzioni per i tuoi problemi.

L'avvocato, il medico, l'industriale, l'artigiano e il negoziante. Tutti oggi ci troviamo spesso di fronte ad una serie di esigenze fiscali, legali, contabili e amministrative, senza contare quelle organizzative e pratiche, che ci portano via sempre più tempo in fastidiosi lavori di routine.

Fortunatamente oggi c'è Harden.

Harden non si limita a consigliare e a vendere il computer più adatto e più conveniente in rapporto a ciascuna esigenza, sia come dimensione che come marca (è esclusivista per l'Italia della Commodore, della Compucorp e OEM Data General) ma provvede anche all'addestramento di chi dovrà usare la macchina, alla manutenzione e all'assistenza tecnica, nonchè a qualsiasi esigenza di software, sia con migliaia di programmi già sperimentati e collaudati sia preparando programmi specifici su misura.

Venite di persona, scriveteci: ci sono più di 400 punti di vendita e assistenza Harden, in Italia.

HARDEN

PIEMONTE E VAL D'AOSTA: Tel. 011/389328-332065 ● LOMBARDIA: Tel. 02/4695467 ● VENETO: Tel. 0444/563864 ● FRIULI V. GIULIA: Tel. 040/793211 ● UDINE: Tel. 0432/291466 ● TRENTINO A.A.: Tel. 0471/24156 ● LIGURIA: Tel. 0185/301032 ● EMILIA ROMAGNA: Tel. 0544/30258-30081 ● TOSCANA: Tel. 055/63396 ● MARCHE: Tel. 071/896907 ● UMBRIA: Tel. 0761/224688 ● LAZIO: Tel. 06/8272415 ● ABRUZZI: Tel. 085/50883 ● CAMPANIA: Tel. 0824/24168-21680 ● PUGLIE: Tel. 0881/76111 ● BASILICATA: Tel. 080/481327 ● CALABRIA: Tel. 0984/71392 ● SICILIA: Tel. 090/2928269 ● SARDEGNA: Tel. 070/663746